11-2021

A Comprehensive Approach to Automated Sign Language Translation

Tejaswini Ananthanarayana
ta2184@rit.edu

A Comprehensive Approach to Automated Sign Language Translation

by

Tejaswini Ananthanarayana

A Dissertation Submitted in Partial Fulfillment of
the Requirements for the Degree of
Doctor of Philosophy in Engineering

Supervised by

Dr. Ifeoma Nwogu
Department of Computer Science
Golisano College of Computing and Information Sciences

Rochester Institute of Technology
Rochester, New York 14623
November 2021

Approved By:

Dr. Raymond Ptucha,
Thesis Advisor, Department of Computer Engineering, Kate Gleason College of Engineering

---

Dr. Majid Rabbani,
Committee Member, Department of Electrical and Microelectric Engineering, Kate Gleason College of Engineering

---

Dr. Alexander Loui,
Committee Member, Department of Computer Engineering, Kate Gleason College of Engineering

---

Dr. Andreas Savakis,
Committee Member, Department of Computer Engineering, Kate Gleason College of Engineering

Acknowledgments

Throughout my Ph.D. journey, I am grateful to many people who have contributed in different ways towards my success.

First and foremost, I would like to thank my advisor, Dr. Ifeoma Nwogu, for graciously accepting me mid-way during my Ph.D. I was able to complete my Ph.D. successfully only due to the constant guidance and support provided by her. Although we celebrated our success during different paper and journal publication acceptances, I think we bonded more during the rejections. Dr. Nwogu not only lifted my hopes but also motivated and encouraged me to never give up and continue trying by enhancing my work even further. I have learned so much from her, not just technically but also non-technically. I feel so honored to have worked with her and to be her student.

The foundation that I have in deep learning is all thanks to Dr. Ptucha, my second advisor. I feel privileged to have started my AI journey with him. I am so grateful to Dr. Ptucha for exposing me to so many industrial projects and coops. I would have never gained the experience I have today if it were not for the opportunities that he presented.

My deep learning gurus, Dr. Nwogu and Dr. Ptucha have pushed me towards success and have supported me through this entire journey. I can't thank both of them enough!

I would like to thank all my committee members, Dr. Andreas Savakis, Dr. Majid Rabbani, Dr. Alex Loui, and Dr. Ray Ptucha for making time for my defense and for providing me with valuable feedback and reviews that have helped me in improving my work. I would also like to thank, Dr. Sonia Lopez Alarcon, for identifying my potential in doing a Ph.D.

My deepest gratitude goes out to Dr. Edward Hensel. Dr. Hensel laid out a well-structured and detailed itinerary for my Ph.D. journey. He has always had my back and has completely supported

me throughout my Ph.D. career. I would also like to take a moment and thank Rebecca Ziebarth for taking care of all the formalities diligently and answering my questions patiently.

During my Ph.D. career, I have been fortunate to work with a couple of companies and wonderful teams. I would like to thank Bob Childs and his team from Synaptics Inc for providing me with a very good digital design experience. I would like to thank Sean Kelly, my manager from ON Semiconductor for allowing me to contribute to the deep learning arena at ON.

I would like to thank Shagan, Miguel, Lipisha, Nicholas, Priyanshu, Nikunj, Milind, Joe, Akash, Akhil, Brian for their contribution. I would like to especially thank the CE IT department, specifically Richard Flegal for all the support provided towards MIL workstations.

Special thanks to my husband, Aditya Giriyalkar, for he has supported, encouraged, loved, and tolerated me like no one else ever has. I am forever indebted to my parents (mom (Vasu) and dad (Ananthanarayana)) for making me who I am today. I would like to thank my grandparents-in-law (Shrikant & Vimal), my father-in-law (Ashok), mother-in-law (Archana), brother-in-law (Aniket) for their constant emotional support and encouragement. Last but not the least, I am forever grateful to my kids (dogs - Rocky & Sandy) for loving me unconditionally.

This thesis is dedicated to my mom (Vasu).

Abstract

Many sign languages are bonafide natural languages with grammatical rules and lexicons, hence can benefit from neural machine translation methods. As significant advances are being made in natural language processing (specifically neural machine translation) and in computer vision processes, specifically image and video captioning, related methods can be further researched to boost automated sign language understanding. This is an especially challenging AI research area due to the involvement of a continuous visual-spatial modality, where meaning is often derived from context. To this end, this thesis is focused on the study and development of new computational methods and training mechanisms to enhance sign language translation in two directions, signs to texts and texts to signs.

This work introduces a new, realistic phrase-level American Sign Language dataset (ASL/ ASLing), and investigates the role of different types of visual features (CNN embeddings, human body keypoints, and optical flow vectors) in translating ASL to spoken American English. Additionally, the research considers the role of multiple features for improved translation, via various fusion architectures. As an added benefit, with continuous sign language being challenging to segment, this work also explores the use of overlapping scaled visual segments, across the video, for simultaneously segmenting and translating signs.

Finally, a quintessential interpreting agent not only understands sign language and translates to text, but also understands the text and translates to signs. Hence, to facilitate two-way sign language communication, i.e. visual sign to spoken language translation and spoken to visual sign language translation, a dual neural machine translation model, SignNet, is presented. Various training paradigms are investigated for improved translation, using SignNet. By exploiting the notion of similarity (and dissimilarity) of visual signs, a metric embedding learning process proved most useful in training SignNet. The resulting processes outperformed their state-of-the-art counterparts by showing noteworthy improvements in BLEU 1 - BLEU 4 scores.

Contents

List of Figures

List of Tables

1. Overview and Motivation

According to the World Health Organization, approximately 430 million people have hearing loss and require some rehabilitation assistance [4]. Sign language (SL) is a natural visual-spatial language used for communication among many Deaf and Hard-of-Hearing (DHH) people. It is estimated that there are over 200 [5] different sign languages used around the world and sign languages are not necessarily mutually intelligible even if their spoken counterparts are similar[6].

Unfortunately, the majority of hearing people do not understand sign language and this may result in missed opportunities for some members of the DHH communities, in areas such as education, employment, sports, and other social activities. Sign language interpretation can address many of these challenges by facilitating the communication between signers and non-signers. But the use of human interpreters for sign language translation can be inconvenient, difficult to schedule, and costly, hence more research groups are attempting to automate this translation process using machine learning techniques. It is important to note that an artificial intelligence (AI) tool such as we propose facilitates the language system support required by both DHH signers and non-signers, but is not a replacement for sign language interpreters or direct signer-to-signer communication.

Unlike regular hand gestures, sign languages typically have five parameters (grammatical features): handshape, location (use of space), palm orientation, body movement, and facial grammar (non-manual signals) [7, 8]. Several of these five parameters such as leaning the body forward/backward, head turns and shakes, eyebrow movements, nose wrinkling, mouth movements, etc., do not have equivalents in written or spoken language grammar. For this reason, unlike their spoken/written language counterparts, for machine analysis, sign languages need to be encoded visually and spatially. Deep learning based computer vision techniques, provide a framework to support this.

The unit components of sign language are not directly translatable to their spoken word counterparts; rather, signs are directly related to an intermediary symbolic language called *gloss*. Gloss can be

viewed as the written form of sign language and is useful for transcription. More information on gloss is provided in Section 1.2. As glosses align directly with signs (unlike spoken words), initial studies in automated SL understanding required them as an intermediate step in performing translation, a problem we refer to more appropriately as recognition or transcription. The availability of gloss simplifies the task of SL understanding, as the model first performs gloss-based recognition and as a secondary step, translates the gloss to text [9]. Alignment-based loss functions such as the connectionist temporal classification (CTC) loss are used and they go a long way in facilitating the process.

1.1 Motivation

Research on automated sign language understanding has been limited, especially when compared with its spoken language counterpart. Sign language interpretation can be challenging due to the lack of easy accessibility to human interpreters - sign language interpreters can be expensive and not readily available. Automating the process of sign language translation (SLT) can therefore greatly facilitate the communication between signing and non-signing people in the community. Additionally, for real-time sign language understanding in real-world situations, where gloss is unavailable, it is imperative to design SL systems that can bypass the need for gloss during translation.

To this end, this thesis is focused on the study and development of new computational methods and training mechanisms to enhance sign language translation in two directions, signs to texts and texts to signs. We initially explain glossing and its caveats in detail in Section 1.2. In the SL research world, transcription (recognition) and translation have sometimes been used interchangeably but hold different meanings. We discuss the distinctions between transcription (recognition) and translation in Section 1.3. Most state-of-the-art (SOTA) translation work use gloss as an intermediate step or used gloss-based features (explained in detail in Section 1.4). Since gloss is expensive to get our sign to text translation work majorly focuses on bypassing gloss in every way possible to make our models robust and adaptable to realistic scenarios.

Many sign languages are bonafide languages with their own grammars and lexicons [10], hence there are rules governing the order of signed words along with the forms of signed words to use

when communicating. Because of these underlying rules, we can implement machine translation methods for sign languages. Throughout this thesis, we study the effects of different sign languages on the deep learning models for SLT. Chapter 2 discusses all the SL datasets used for evaluation across various methodologies. We discuss how these datasets were collected and introduce a new dataset collected in realistic settings and environments.

We discuss how SLT has evolved from sequence modeling for text-to-text translation to modern transformer-based translations in Chapter 3, Section 3.1. To find the best model that would serve as our base model for SLT we evaluate various deep learning based methods for *encoding* sign language at the front-end and combine these with various deep learning based *machine translation* techniques at the back-end. Specifically, for machine translation, we implement sequence-to-sequence models with and without attention, a reinforcement learning (RL) based model, and a transformer model for sign language to text translations. These methods are discussed in detail in Chapter 3, Section 3.2. To the best of our knowledge, this is the first work to perform continuous sign language translation, without the intermediary gloss based recognition step.

To understand the role of the different input features discussed in Chapter 4, Section 4.1 we perform ablation studies over the model architectures (input features + neural translation models), for improved continuous sign language translation. These input features include body and finger joints, facial points, as well as vector representations/embeddings from convolutional neural networks. We implement the translation methods over multiple sign languages - German (GSL), American (ASL), and Chinese sign languages (CSL). Training details and performance analysis of these methods are discussed in Chapter 4, Section 4.2 and Section 4.3 respectively.

Using the learning's from Chapter 3 and Chapter 4 we introduce a new, realistic phrase-level ASL dataset (ASLing), and explore the fusion of visual, spatiotemporal, and pose-based features in sign language translation, by proposing two different architectural models. Chapter 5 explains these novel fusion models in detail. We perform experiments to show how SLT benefits from our dynamic fusion models when compared to single feature counterparts. Sign language is challenging to segment, we, therefore, obtain the input features by extracting overlapping scaled segments across the video and obtain their 3D CNN representations. We exploit the attention mechanism by choosing the best fusion architecture. In Chapter 5, Section 5.4 we initially learn dependencies between

different frames of the same video and later fusing them to learn the relations between different features from the same video using scaled 3D CNN features and pose-based features.

Much of the AI work in sign language translation to date has focused primarily only in one direction, going from signs to text. This puts the advantage mainly on the side of the hearing-centric who receive information in their natural language (text/speech), but this is not the case for the Deaf-and-Hard-of-Hearing (DHH) whose natural mode of communication is sign language. To this end, in Chapter 6, SignNet, with two training paradigms is presented, a coupled model and a decoupled one, both facilitating two-way communication in sign language (visual sign to spoken language translation, and spoken to visual sign language translation).

1.2 Glossing and its Caveats

Glossing is a written form of sign language [11, 12], though it does not have the same structure as the spoken language equivalent. When a person signs a phrase, glossing refers to the written form of the sign-for-sign transcription (not translation) of that phrase. For example when signing the phrase "My name is Albert", the signer might finger-spell "Albert" and conceptually place that spelled construct at some location in a 3-dimensional space in the vicinity of the signer. This way, any ensuing references to the name "Albert" will no longer require full finger-spelling, rather the signer can point to that abstract location in space where the construct was stored. While the gloss can accurately capture this process, there is no equivalent representation in the spoken/written equivalent language. Sign language translation aims more to preserve the overall meaning of the phrase while following the rules of the language grammar. As another example, American Sign Language (ASL) makes use of special signs known as *classifiers*. A classifier categorizes a bunch of phrases to a single sign which then can be used as a reference while signing.

An example is the "3-handshape" used to represent a vehicle. But the sign can also include the orientation, speed, location, and direction of travel of the vehicle. The written form or gloss of a sign includes information about these classifiers and also includes information relating to facial and body movements.

Some glossing terminologies include information like `LOC` referring to a location, `PO` referring

Table 1.1: Glosses (sign-for-sign transcriptions) and their spoken sentence translations from the ASL dataset [2].

No.	Gloss	Spoken Sentence	Selected gloss meaning
1.	IX-loc:j OVER/AFTER EXAGGERATE IX-3P:j LATER_2 USE COAT RAIN COAT USE UMBRELLA BOOT PANTS_3	People go too far: they use umbrellas, wear rain coats, and put boots and pants on.	IX: index, IX-loc: points to a location.
2.	IX-1p SAY part:indef UP-TO-NOW IX-3p:I 5"looking for words" IX-3p:I fs-PAUL KNOW IX-3p:I POSS-1p (1h) GOOD/ THANK-YOU FRIEND	I said to Paul in the email, "you know, you have been a good friend of mine"	fs: fingerspell, POSS-1: Possessive pronouns (my, mine, etc,).
3.	IX-1p LOOK:p 5"resignation" FINE IX-1p WAIT++	Thought "ugh" and ended up waiting again.	++: repeat sign.

to palm orientation, `LCL` referring to locative classifiers, etc. Glossing also includes information based on the number of times a sign is repeated. For example, the plus sign ++ at the end of a gloss indicates a number of repetitions of an ASL word; e.g. `again++` (signing the word “again” two more times) means “again” and “again”. Another example, `HELP+++` could mean “help many times” or “help from time to time”, depending on the context, the duration of the movement, and spatial reference to convey different meanings. Some glosses along with their meanings and spoken sentence equivalent are shown in Table 1.1.

1.3 Translation vs Transcription

Completely transcribing sign language gestures (or signs) into written language or attempting to use only written language to fully represent signs is an extremely challenging task. Many signs incorporate movement and space (within the sign) to modify the meaning. It is therefore challenging to encapsulate such spatially oriented information into words.

Also, much of sign language grammar is conveyed specifically by facial and body movements, not present in written texts, thereby rendering them even more challenging to encapsulate. For these

reasons, sign languages are often transcribed first into an intermediary written representation called *gloss*, which captures both the sign-for-sign word ordering and the different notations needed to account for spatial-temporal, facial, and body grammar.

Sign language *recognition* or *transcription* involves the process of converting signed visual phrases into gloss, whereas sign language *translation* involves going directly from signs to the spoken version of the language. Unlike many recent works in sign language analysis [9, 13], where recognition is used as an added step to boost translation, in this work, we focus strictly on the challenging task of translation only, especially when gloss is not available as is the case for many low-resource sign languages.

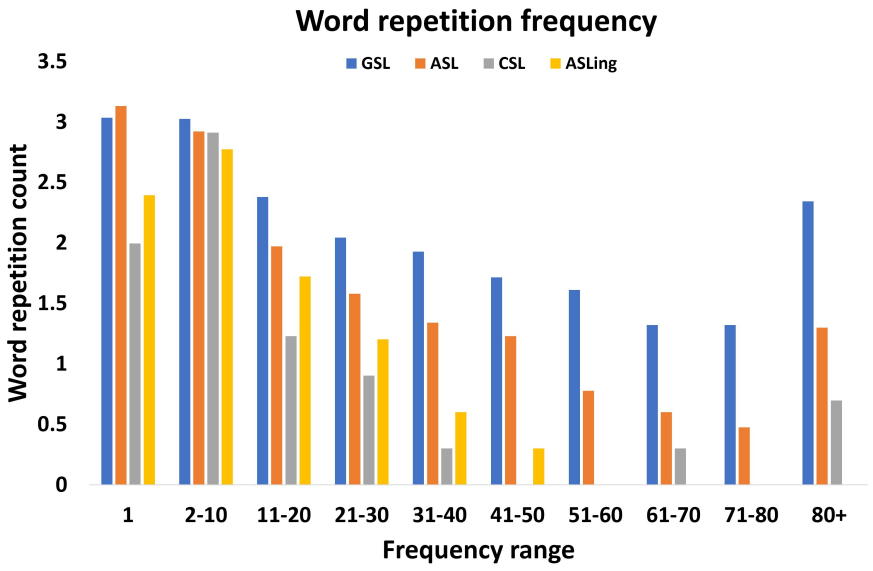
1.4 Gloss-aligned visual features

In this section, we briefly review one of the more successful input features that have been used for sign language understanding. The model used by several researchers [14, 13], yielding good metrics on benchmark datasets are based on a CNN-LSTM-HMM model [15]. The CNN-LSTM (Convolution Neural Network, Long Short-Term Memory) part is initially trained as a classifier in a weakly supervised manner to identify the gloss based classes, hand, and mouth shapes; then the probabilities from the CNN-LSTM model are fed to a hidden Markov model (HMM), further used for alignment. This CNN model is then initialized with the pretrained weights and used to extract features for the sign language video under consideration. Hence gloss is used also in the feature extraction process.

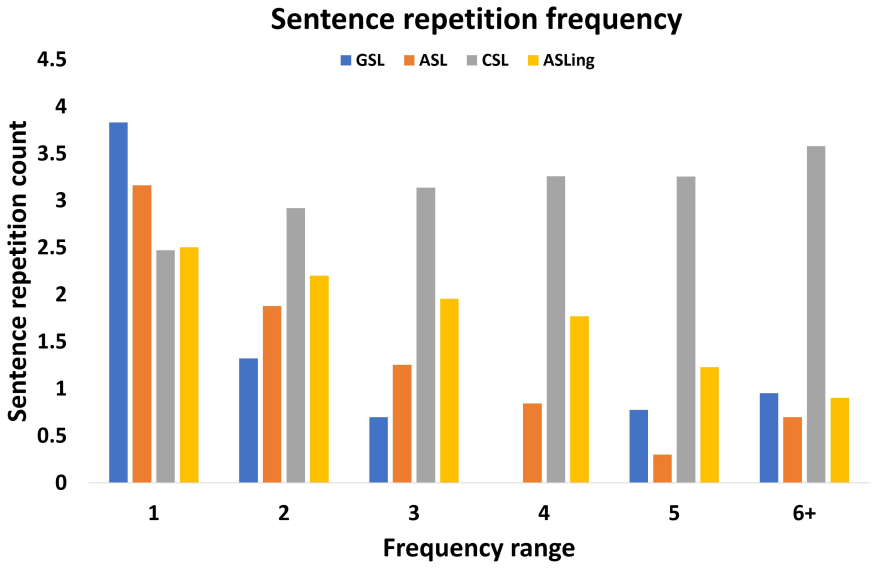
2. Datasets

Throughout the thesis, we explore different sign language datasets and evaluate our models on these datasets. The environment in which data was collected, the quality of the data, the category of topics covered, etc have a powerful impact on the performance. This section highlights all these details on the different datasets used in this work. Some examples of frames from these datasets are shown in Fig. 2.1 to illustrate the nature of the input data we are working with.

Figure 2.1: Random samples from the datasets: top to bottom - GSL dataset [1], ASL dataset [2], CSL dataset [3], and ASLing dataset.



(a)



(b)

Figure 2.2: (a) Word repetition frequency for unique utterances. (b) Sentence repetition frequency. Y-axis represents the log10 scale.

2.1 German Sign Language (GSL)

This data is collected from weather forecast airings from the RWTH-PHOENIX-Weather dataset [1]. The videos were recorded at 25 frames per second with 210×260 pixels as the frame size. Fig. 2.2 (a) highlights the word level frequency for unique utterances. We use the same train/dev/test split like the original creators of the GSL dataset [1]. The GSL dataset signed by nine signers collected 7096 training videos and ground truth captions out of which 6811 are unique utterances as seen from the sentence repetition frequency in Fig. 2.2 (b). There are 1078 unique words in the dataset with 1042 repeated $2-10$ times and other words spread out with frequencies greater than 11. The dataset is also confined to only weather-related sentences, thus spanning a confined subject area as shown in the word cloud in Fig. 2.4. Fig 2.3 shows that the majority of the annotated sentences are signed by signer 1, signer 4, and signer 5 with an unequal distribution of sentences between the nine signers. The GSL dataset was collected in a controlled environment with a similarly illuminated background for almost all the videos with color contrasts in the clothes worn by the signers as seen in the top row of Fig. 2.1. The dataset is also confined to only weather-related sentences, thus spanning a confined subject area as shown in the word cloud in Fig. 2.4.

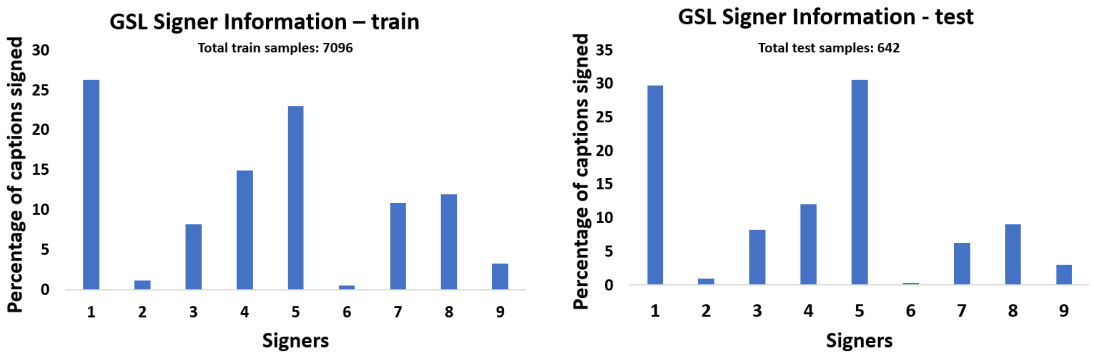


Figure 2.3: Graph representing number of signs annotated by each signer in the GSL dataset.

2.2 American Sign Language (ASL)

The American Sign Language (ASL) dataset [2] consists of videos narrating 38 different topics in American sign language. The frames were extracted at 25 frames per second with the resolution of videos ranging from 216×218 to 312×324. The extracted dataset has videos signed by 7 signers.

Figure 2.4: Word clouds for GSL, ASL, CSL, and ASLing datasets.

The distribution between the signers is shown in Fig. 2.5. The dataset consists of 38 stories. As the videos are approximately 1.5 to 2 minutes long on average, each video is divided into multiple sub-videos based on individual utterances. Due to this, some of the videos and utterances do not match one-to-one and the quality of the sub-videos are also affected poorly. The ASL dataset has around 1457 unique utterances. The word and sentence repetition frequency are shown in Fig. 2.2 (a) and Fig. 2.2 (b) respectively.

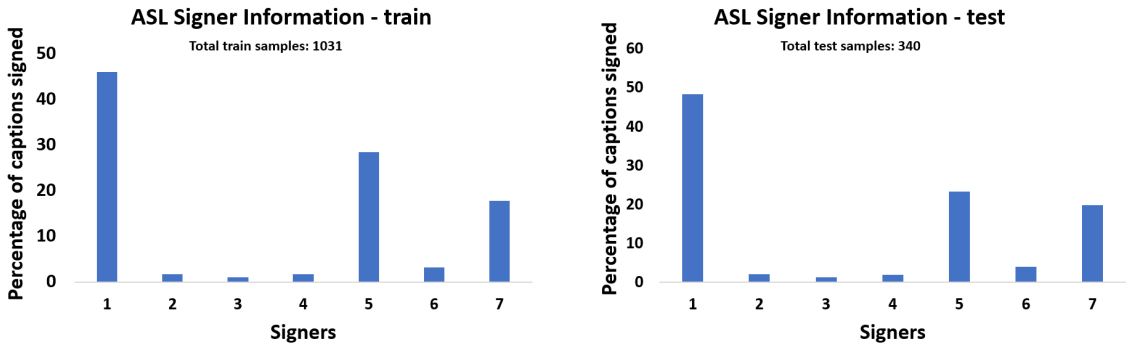


Figure 2.5: Graph representing number of signs annotated by each signer in the ASL dataset.

From Fig. 2.1 middle row, it can be seen that very similar to the GSL dataset, the ASL dataset also has a controlled background with contrast in cloth colors. The signers appear to be signing professionally with ease.

2.3 Chinese Sign Language (CSL)

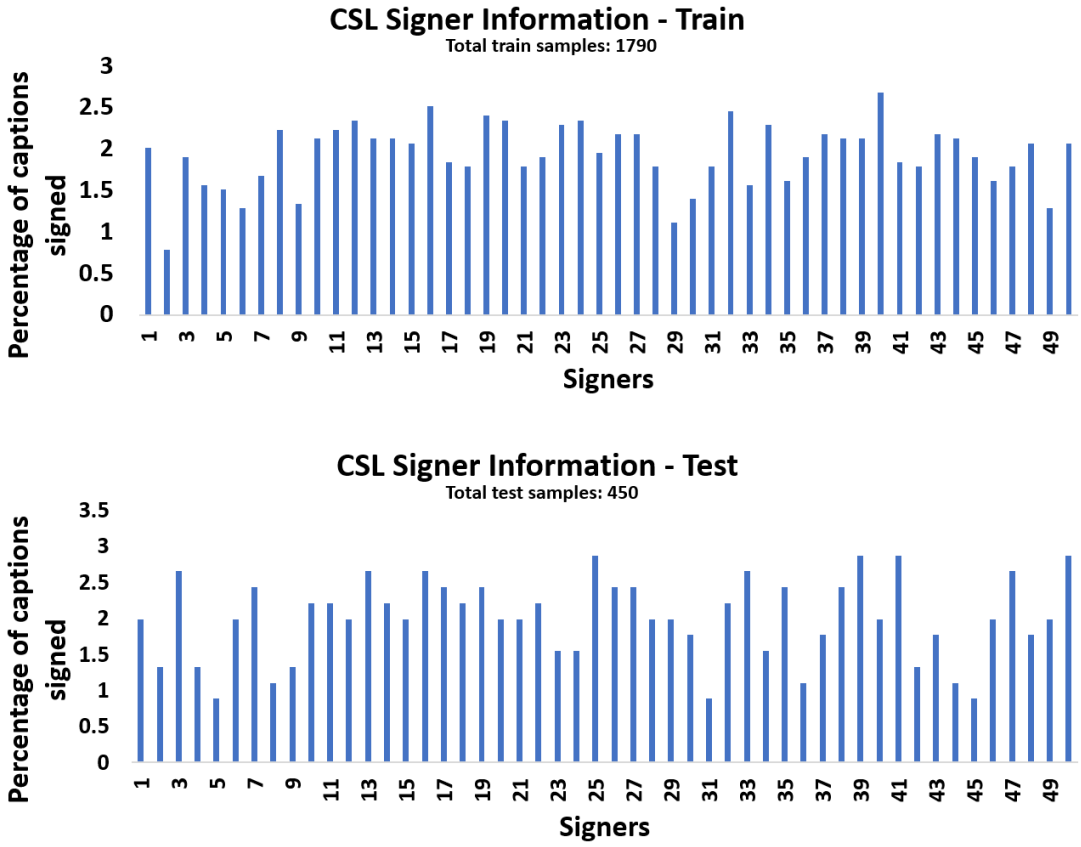


Figure 2.6: Graph representing number of signs annotated by each signer in the CSL dataset.

The Chinese Sign Language Dataset [3] consists of approximately 50,000 Chinese sign language videos. 1000 utterances were assigned to 50 Deaf signers (25 male and 25 female). All signers have a fair distribution of utterances assigned for signing as shown in Fig. 2.7. 10,000 utterances have been signed in the CSL dataset. The videos were recorded at 30 frames per second. The color image frames are 1920×1080 pixels in size. The segmentation of words and characters specific to the Mandarin language was done using the Chinese word segmentation tool, Jieba[1]. Fig. 2.2 (a) shows the word-level frequency among unique utterances. Fig. 2.2 (b) shows the repetition frequency of the 10,000 utterances. This CSL dataset was based on a recently standardized sign language in China [3] and hence the signers used in this dataset are not native signers. They were

[1]Jieba Chinese text segmentation. https://github.com/fxsjy/jieba/

provided training videos showing experts performing these new standardized signs, and the signers mimicked the videos by experts. These videos are taken in different settings and backgrounds and at different positions from the camera.

2.4 American Sign Language (ASLing)

The American Sign Language (ASLing) dataset consists of 1027 training and 257 testing samples. We interchangeably use ASLing and ASL, both refer to the same dataset in our work. These videos were collected at 10 frames per second and were annotated by 7 signers. Each frame is of 450×600 size. The ASL dataset consists of a wide variety of topics, unlike GSL that is more constrained on weather-related topics.

Although ASLing is currently the largest phrase-based ASL corpus, the data is very noisy (collected in real-life settings) and is thus challenging to analyze. The presence of poor illumination during collection results in low-quality images, making the dataset even more challenging to analyze. These issues are in contrast with the GSL dataset which was collected under significantly more controlled conditions.

The word cloud in Fig. 2.4 (a) shows the organization of different words in the dataset. Note the variation in ASL word topics compared to GSL. The word and sentence repetition frequency for the ASLing dataset is shown in Fig. 2.2.

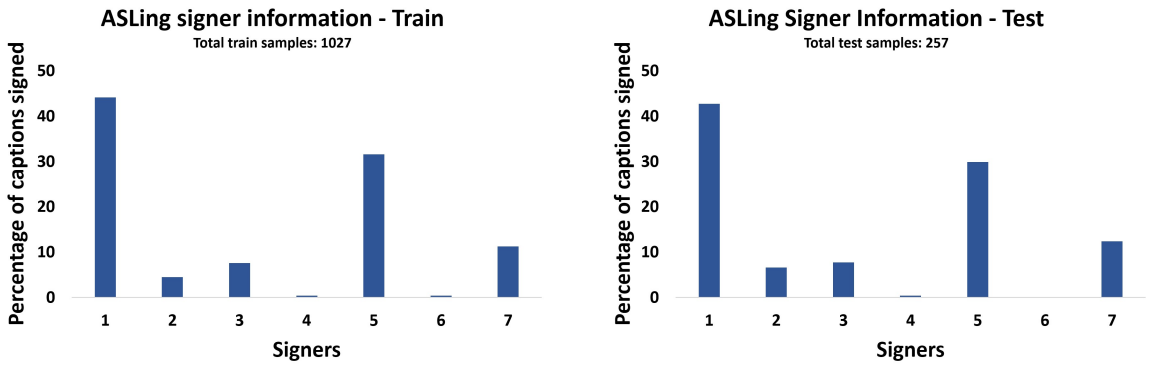


Figure 2.7: Graph representing number of signs annotated by each signer in the ASLing dataset.

To understand the distribution of the two datasets, we converted the German words to English and

projected the word2vec embeddings using PCA for both the GSL and ASL dataset as shown in Fig. 2.8. The ASL (shown in red) dataset is widely spread out whereas the GSL (shown in blue) dataset covers a very confined subject.

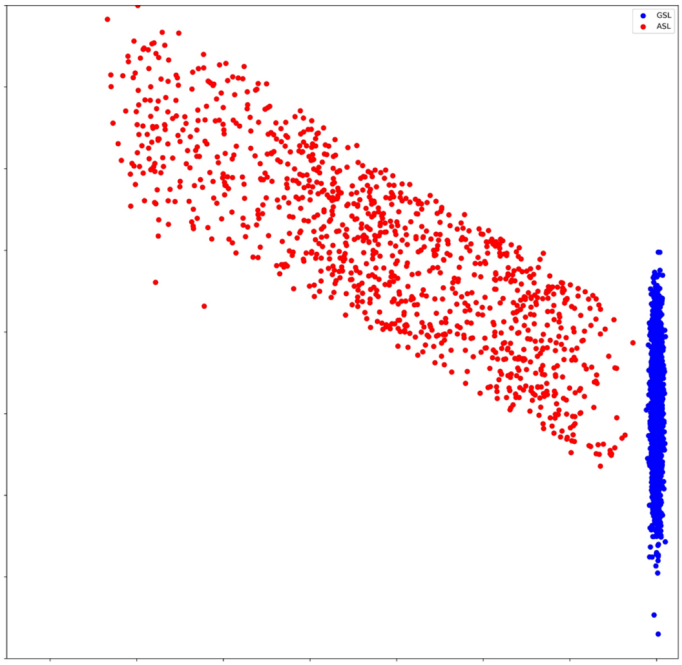


Figure 2.8: Embedding space for the GSL and ASLing datasets using PCA (best viewed in color) (Red - ASLing, Blue - GSL).

3. Technical Background

In this chapter, we discuss research and study done towards the field of SLT along with preliminary background work done to lay a strong foundation for my thesis.

3.1 Related Works

Sign language translation borrows concepts from various natural language processing (NLP) tasks where sentences from one language are translated to the other. In the following subsections, we discuss some of these methods and highlight existing research being done that contributes in one way or the other towards sign language interpretation. In the following subsections we discuss different sequence modeling techniques related to sign language understanding including text-to-text translation, video captioning, gesture recognition, reinforcement learning, and video sign language to text translation.

3.1.1 Sequence modeling for text-to-text translation

One application of sequence modeling is machine or language translation. Neural machine translation is the task of translating a text from one language to the other. Machine translation tasks started to gain recognition from the work proposed by *Kalchbrenner and Blunsom* [16] where a probabilistic continuous translation model was introduced. Following this work, *Sutskever et al.* [17] introduced a multilayered LSTM [18] to take in a variable number of encoded words of the input sentence and translate it to a target language using another LSTM in the decoder. The performance of these networks was limited by the length of the input sequence which was highlighted by *Cho et al.* [19]. *Bahdanau et al.* [20] extended LSTMs by introducing attention mechanism. This involves taking a linear combination of encoder hidden states when predicting the next target word in the decoder. There are many other works using attention mechanisms for the neural machine learning tasks [21, 22].

Vaswani et al. [23] introduced a text-to-text based model completely based on attention and feed-forward networks without the use of RNNs. These models, called transformer models, process all input words in parallel using the query, key, and value word embeddings. Other works such as Generative Pre-Traning models, GPT-1 [24], GPT-2 [25], and GPT-3 [26] have extended the work by *Vaswani et al.* These models are designed for text-based tasks like text classification, sentence similarity, question answering, next-word prediction, and text summarization. These GPT models initially train the transformer model with large unlabeled data in an unsupervised fashion and later perform supervised learning to address the above-mentioned tasks. GPT-3 [26] is a very big language model with 175 billion parameters. Bidirectional Encoder Representations from Transformers (BERT) [27] model, also inspired by the transformer model, trains on unlabeled data in an unsupervised manner and these pre-trained weights are used to fine-tune on labeled data for downstream tasks like question and answering.

3.1.2 Video captioning

Sequence modeling has also been extended towards other variable length tasks like video description, visual question and answering, scene description, and action or gesture recognition. *Venugopalan, et al.* [28] introduced a sequence-to-sequence video to text model with two stacked LSTMs. Each frame is passed one at a time during the encoding stage. The hidden representation from the encoding stage is passed onto the decoding stage where the model predicts one word at a time. Hierarchical Recurrent Neural Networks proposed by *Yu et al.* [29] not only include attention mechanisms but also use sentence and paragraph generators. Researchers have explored different variations of LSTMs such as leveraging attention mechanism with semantic consistency [30, 31], and LSTMs with temporal modifications based on the discontinuities between the frames [32]. *Olivastri* [33] perform end-to-end captioning of videos with an encoder-decoder architecture using LSTMs with attention. *Aafaq et al.* [34] performed comprehensive ablation studies for video captioning across several datasets and compare benchmark results based on the size of the dataset and number of classes. They evaluate results on different evaluation metrics to highlight the advantages and disadvantages of different metrics. RL based video captioning is also gaining popularity

with a teacher-student module [35, 36]. To enable processing of longer sequences, transformer-based models [37, 38, 39] and BERT-based models [40, 41] are also contributing towards improving the performance of the video captioning tasks. Vision-and-Language BERT (ViLBERT) [41] introduced co-attentional transformer layers over two parallel pipelines, one for text and the other for visual data. The co-attentional layer helps in providing information from the video pipeline to the text pipeline and vice-versa. Transformer-XL [42] enhanced the NLP tasks by learning even longer-term dependencies in a computationally contractible fashion.

3.1.3 Gesture recognition

Gesture and action recognition is a variant of sequence modeling. These are often treated as categorical classification problems. In these types of models, the temporal features of the video and specifically the motion of the body, hand, and face play a major role. *Karpathy et al.* [43] introduced several early, late, and mid-fusion temporal variations of convolutional neural networks (CNN) for video classification. *Pigou et al.* [44] and *Nishida et al.* [45] used recurrent architectures for gesture recognition. To facilitate the gesture recognition process, several works have studied ways to extract 2-D and 3-D locations of the body, hands, and face. OpenPose [46, 47, 48, 49] consolidated many of these works into a single framework by providing 70 facial keypoints, 25 body keypoints, and 21 hand key points for each hand. OpenFace [50] extracts facial landmark and action units. Artrack [51] also provides key joints in the body. While gesture and action recognition analyze a sequence of frames for discrete classification, continuous sign language translation analyzes a stream of frames, continuously outputting textual content that takes into consideration prior video context.

3.1.4 Reinforcement Learning

Reinforcement learning (RL) has been very popular in robotics research [52, 53, 54, 55]. RL gained a lot of fame from its successes in playing the well-known *Atari* game [56] and *AlphaGo* game [57, 58]. However, only recently has it started to contribute to the field of sequence modeling. A typical sequence-to-sequence model like LSTM is trained such that it predicts the next word based on the previous ground-truth word. This method is also sometimes referred to as *"teacher forcing"* where instead of passing the previous predicted word to obtain the next word, the previous ground-truth word is passed. *Teacher forcing* is applied only during training. During test time

the predicted previous word is passed on to guess the next word. This leads to a performance hit, known as *exposure bias*, as the model is not able to generalize well. *Exposure bias* ([59, 60]) leads to error during test time. In addition to *exposure bias* there can be issues because there is no back-propagation of the evaluation metric during testing *i.e.,* we can see *non-differentiable* issues during testing. These two issues are addressed largely by using RL.

Rennie et al. [61] implemented image captioning using self-critical sequence training (SCST) which is based on the *REINFORCE* algorithm [62]. SCST with REINFORCE baselines its experiences from the test-time inference algorithm output to normalize the rewards. A positive weight is assigned to the samples that performed well during test-time inference and those that do not perform well are suppressed. In this work, SCST with REINFORCE was performed after the system was trained with cross-entropy loss for a minimum of 20 epochs. To improve the test-time inference, the Consensus-based Image Description Evaluation (CIDEr) [63] metric is used with SCST along with greedy decoding. CIDEr score is evaluated by matching predicted sentences with the consensus of a set of human-annotated ground truth sentences. *Shi et al.* [64] use two networks, a "policy network" and "value network", which perform joint learning to predict the next word in an image captioning task. While the policy network evaluates the confidence of predicting the next word, the value network evaluates rewards based on the predictions. The policy network, in short, is a supervised network trained with cross-entropy loss. The value network is trained to minimize the mean squared loss. These two networks are then jointly trained to maximize the rewards. *Li et al.* [65] and *Zhang et al.* [60] extend the concept from [61] by using the REINFORCE algorithm towards video captioning and continuous sign language recognition tasks respectively. While [65] uses a sequence-to-sequence LSTM based network with the CIDEr score, [60] uses the transformer model and Word Error Rate (WER) to minimize the loss and maximize the reward.

3.1.5 Video Sign language translation

Word level and sentence level datasets are widely used for sign language translation. Word level sign language translation tasks can be classified under image classification or gesture/action recognition. [66, 67, 68, 69] made contributions towards tasks like translating discrete signs, mouth, and hand shapes while *Ye et al.* [70] used a combination of RGB, optical flow, and depth information for

American sign language (ASL) translation using a 3D CNN to classify words using its respective temporal information from the continuous sign language video. Residual architectures with temporal convolutions and framewise classification were used by *Pigou et al.* [71] to classify 100 signs. Word level sign language interpretation is a simpler task than continuous sign language as the former involves classifying the input clip between a given number of classes. Whereas for continuous sign language, the input clip itself is long enough to need to take care of long-term dependencies and the output is a series of words where every current word depends on the previously predicted words.

Continuous sign language utilizes temporal information and can be considered a sub-set of video captioning. *Fang et al.* [72] introduced DeepASL which uses a hierarchical bidirectional deep recurrent neural network for both word-level and continuous sign language translation. DeepASL is an extensive work showcasing sign language translation and speech recognition for real-time two-way communication. *Wang et al.* [73] introduced a hybrid model where C3D-ResNet features from the input frames are passed into a temporal convolution (TCOV) and bidirectional GRU (BGRU) block. TCOV and BGRU are responsible for short-term and long-term learning respectively. A fusion layer is also used with inputs from both the TCOV and BGRU blocks to learn the complementary relationship between them. Connectionist temporal classification (CTC) [74] loss combines losses from these three blocks to predict the words of the sign language sentence. *Pu et al.* [75] used 3D temporal residual CNNs with CTC post-processing for continuous sign language recognition whereas [76] use CTC loss to train end-to-end models with custom networks targeting certain specific problems known as SubUNets. *Camgoz et al.* [9] used a multi-layer RNN and an encoder-decoder structure for sign language to text translation. This model incorporates encoder-decoder attention to improve the learning process. *Yuan et al.* [3] introduced a Chinese sign language dataset (CSLD) and performed Chinese sign language translation by modeling a two-layer LSTM encoder-decoder based architecture using different body, hand, and facial features from input frames. *Ko et al.* [77] studied different methods for sign language translation using various joint locations and CNN features as input on the Korean sign language dataset. In this chapter, we study sign language translation by not only considering different models but also looking at different datasets to understand how the organization of the dataset affects the performance.

3.2 Preliminary Work

In this section, we review different deep learning captioning methods specifically tailored to perform sign language to text translation. We start by briefly explaining a basic Recurrent Neural Network, then sequence-to-sequence models, transformer models, and inclusion of RL.

3.2.1 Recurrent Neural Networks

Neural networks are well suited for handling tasks that have fixed size input and output vectors. For example, a CNN takes as input an image of predetermined resolution and outputs a vector of probabilities for each output class it has been trained on. For temporal tasks, such as videos, this concept can be extended to take in a fixed number of frames. Recurrent Neural Networks (RNN) extend this concept further by using hidden states. Hidden states allow the output to be a function of the current input and the output from the previous time step. By passing the previous hidden state information, the RNN can make predictions based upon all prior inputs. RNNs are powerful architectures for NLP based tasks like language translation, question and answering, image captioning, and video captioning. Vanilla or basic RNNs can take in an arbitrary number of time steps, but unfortunately, the passing of information over time is subject to vanishing gradients, thus limiting its usefulness to only a few time steps. To remember longer-term dependencies of dozens to even over a hundred time steps, a special type of RNN called Long Short-Term Memory (LSTM) [18] can be used. Using a series of gates and a second hidden state called a memory cell, LSTMs can be used as a replacement for the vanilla RNN. Gated recurrent unit (GRU) [78, 79] is a slight variation of LSTM with fewer gates and no memory cells. The LSTM and GRU RNN methodologies have been used extensively in deep learning applications like image captioning, video captioning, speech recognition, and language translation. Even LSTMs and GRUs are however subject to vanishing and exploding gradient problems on long sequences. Transformer model and RL address these issues and are described in Subsections 3.2.4 and 3.2.5 respectively.

3.2.2 Sequence-to-sequence modeling without attention

A basic sequence-to-sequence model for sign language video to text translation inspired by [17, 80, 28] is described in this subsection. The sign language translation task is modeled to take the

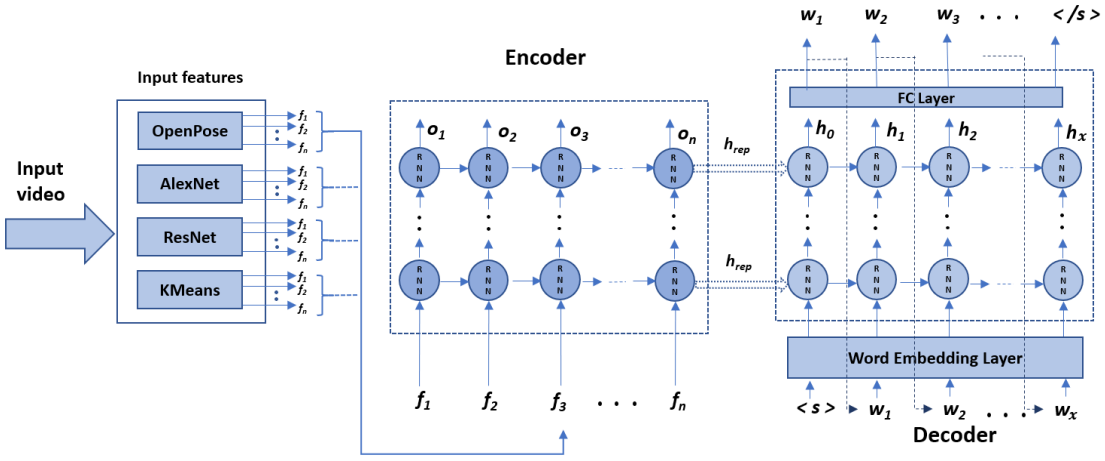


Figure 3.1: Sequence-to-sequence model without attention.

features derived from the input frames such as OpenPose [46], extracted features from a Deep Neural Network (DNN), or KMeans centroids [81]. These features are fed one frame at a time to the encoder. The hidden state from the encoder carries semantically rich features of the signing in the visual domain. The hidden states are propagated to the decoder which employs those features in producing one word at a time. During inference time the decoder starts predicting words upon receiving the "start of the sentence" token <s> and predicts one word at a time. The word predicted at time step t is fed as input to the next layer at time step $t+1$. The encoder and the decoder layers are typically stacked with LSTM units. As this serves as a baseline model for the sign language translation task, experiments using single and multiple stacked recurrent layers on both the recurrent channels have been carried out. The sequence-to-sequence model is shown in Fig. 3.1 with respective notations in Table 3.1.

3.2.3 Sequence-to-sequence modeling with attention

Inspired by sequence-to-sequence models [28, 9] and attention mechanisms [22, 20], an architecture is assembled to perform sign language to text translation as shown in the Fig. 3.2. Inputs to the model are identical to the basic sequence-to-sequence model shown in Fig. 3.1. One frame feature at a time is passed into the encoder. Reversing the sequence order has proven to be beneficial to the model [9, 17] as it helps mitigate long term dependency and vanishing gradients. With respect to Fig. 3.2, $o_{n:1}$ a linear combination of the output of all input time steps and is passed

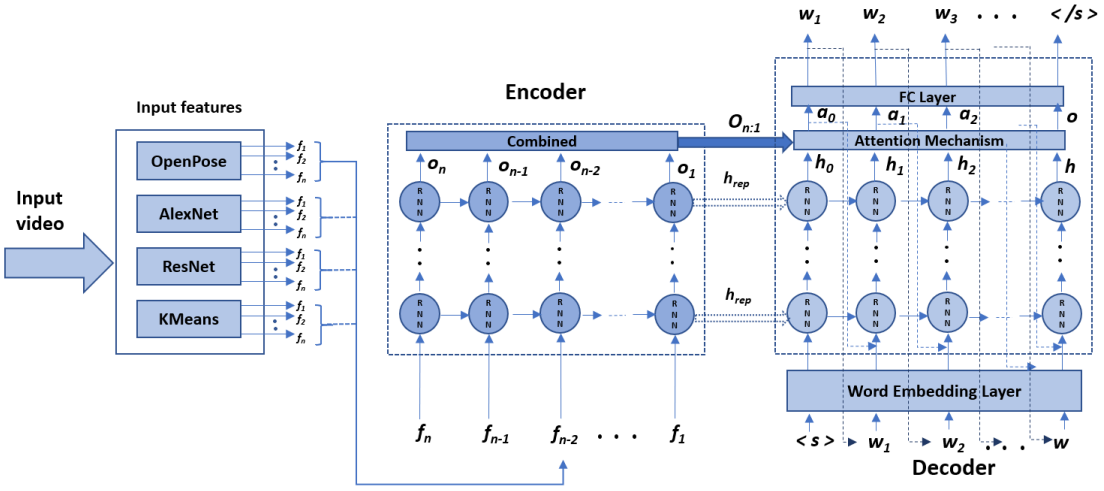


Figure 3.2: Sequence-to-sequence model with attention mechanism.

Table 3.1: Notations for sequence-to-sequence models.

Notation	Meaning
n	Input sequence length
x	Output sequence length
$f_1, f_2, f_3 ....f_n$	Feature vectors for each frame of the video
$o_1, o_2, o_3 ....o_n$	Output vectors from encoder for each frame of the video
h_{rep}	Latent embedding from encoder
$w_1, w_2, w_3 ....w_x$	Predicted words
$a_1, a_2, a_3 ....a_x$	Encoder-decoder attention vectors for each word
$< s >$	Start of sentence token
$< /s >$	End of sentence token

onto the decoder in addition to the latent embedding h_{rep}. The decoder takes in the previous word embedding, the previous hidden state, and the previous attention weights as input. The decoder starts decoding words after receiving the "start of the sentence" token ($< s >$) and predicts one word at a time by feeding the previous predicted word as input to the next time step as explained in Section 3.2.2. The attention mechanism block incorporates encoder-decoder attention by taking in the output vector from the encoder and the previous hidden representation from the output of the decoder. The attention vector is calculated as shown in (3.1).

$$a_x = tanh(W_c c_x + W_h h_x + W_b b) \tag{3.1}$$

where c_x is the context vector, h_x is the hidden state, b is the bias, and W_c, W_h, and W_b are the learned weights for the context vector, hidden state, and bias, respectively. The context vector as shown in (3.2) is a weighted sum of encoder outputs. γ_n^x represents the attention weights.

$$c_x = \sum_{n=1}^{N} \gamma_n^x o_n \tag{3.2}$$

The attention weights highlight the importance of encoder input with the generated word while predicting the next word. The attention weights are typically normalized using the probabilistic softmax equation. Attention weights and mechanisms are shown in (3.3).

$$\gamma_n^x = softmax(f_{att}(h_x, o_n)) \tag{3.3}$$

where, f_{att} represents the attention mechanism used. We give two options for function f_{att}, (3.4a) Luong attention [22] based on multiplication and, (3.4b) Bahdanau attention [20] based on concatenation.

$$X = h_x^T W o_n \tag{3.4a}$$

$$X = V^T tanh(W[h_x; o_n]) \tag{3.4b}$$

Here, W and V are learned weights. Combining (3.1:3.4), the decoder can be expressed as (3.5):

$$y_x, h_x = Decoder(word\_embedding_{x-1}, h_{x-1}, a_{x-1}) \tag{3.5}$$

The decoder stops decoding once it receives the "end of the sentence" token ($< /s >$).

We implement sequence-to-sequence model using LSTM as the RNN by sending in a reversed input frame sequence and incorporating encoder-decoder attention based on Luong attention [22] to predict the words one at a time in the decoder.

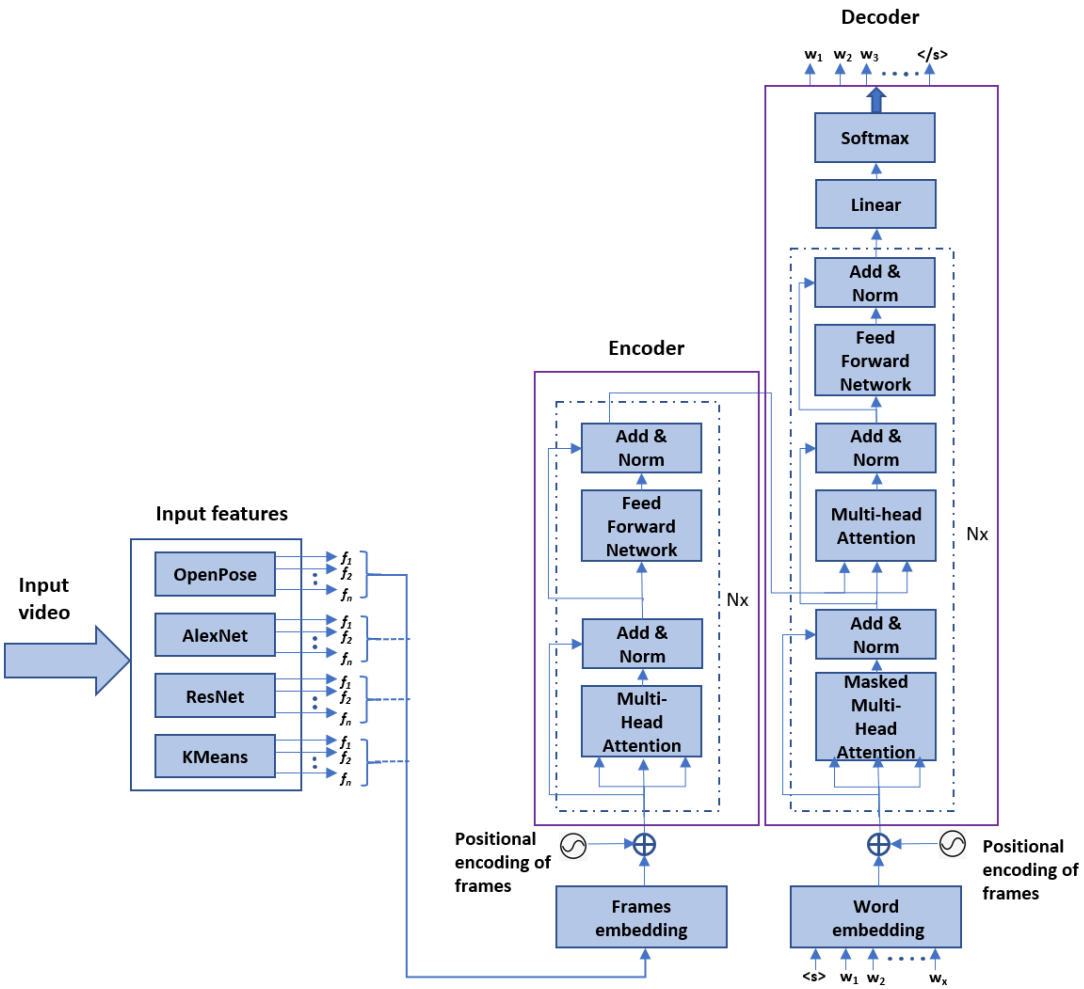


Figure 3.3: The baseline transformer model.

3.2.4 Transformer Model

Transformer models have demonstrated remarkable capabilities lately for language translation tasks [23, 24, 25, 27]. This idea was leveraged for video captioning [42, 37, 38]. Recently, *Camgoz et al.*[13] modified the transformer model for sign language translation using CTC loss. The features used in this model were trained on a CNN-LSTM-HMM architecture [15] where gloss labels are used in a weakly supervised setup followed by an HMM model which performs the alignment. *Yin* [82] experimented with the transformer model by using a different number of layers in the transformer model and perform Sign to Gloss and Gloss to Text translation. We focus on the transformer model shown in Fig. 3.3, which is a state-of-the-art model introduced by *Vaswani et al.* [23] for

the task of sign language to text translation. While an LSTM has been shown to produce good results with over 100 output tokens long, transformers can produce good results with over 1,000 output tokens long. The transformer encoder and decoder layers are repeated *N* times. Each layer in an encoder consists of two sublayers and the decoder consists of three sublayers. Unlike a typical RNN where a sequence of inputs is fed one at a time, the transformer takes all the inputs together. To help the model understand the order of input, a positional encoding parameter is introduced. The positional encoding provides the order information in the sequence of inputs and this is added to the input embedding before passing it to the encoder. Positional encoding is given in (3.6) [23].

$$\begin{aligned} PE_{(pos,2i)} &= sin(pos/10000^{2i/d_{model}}) \\ PE_{(pos,2i+1)} &= cos(pos/10000^{2i/d_{model}}) \end{aligned} \tag{3.6}$$

where pos is the position of the input in the sequence of inputs and i represents embedding dimension. The first sub-layer in the encoder is the "Multi-head Attention". The customized positional encoding with input embedding is now used to calculate the keys (K), queries (Q), and values (V) by learning three distinct linear layers as shown in (3.7).

$$\begin{aligned} K &= linear\_key(input) \\ Q &= linear\_query(input) \\ V &= linear\_value(input) \end{aligned} \tag{3.7}$$

In (3.7), input is a combination of input embedding and positional encoding. *linear_key*, *linear_query*, and *linear_value* are three distinct linear neural network layers for *K*, *Q*, and *V* respectively. These three linear layers can also be considered as a typical feedforward neural network (FFN) layers which are learned separately with three different weights. The attention is then calculated as shown in (3.8) [23].

$$Attention(K, Q, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{3.8}$$

d_k is the dimension of the key vector. Equation (3.8) is scaled by a factor of $\sqrt{d_k}$ to avoid the larger values of dot product which will lead to smaller gradients due to softmax. This attention mechanism is called *Scaled Dot-Product Attention*. The authors [23] introduced the concept of *Multi-Head*

Attention where the linearly learnt queries, keys, and values are projected separately h times where h represents the number of heads. For each of these projections, scaled dot-product attention vectors are calculated as per (3.8), and the results for all the heads are concatenated together in a final linear layer to obtain the multi-head attention layer output. This multi-head attention output is passed through a feed-forward network (FFN) with two linear layers and a ReLU activation as shown in (3.9), where x is the output from multi-head attention block,

$$FFN(x) = max(0, xW1 + b1)W2 + b2 \tag{3.9}$$

$W1$ and $b1$ are the weights and biases respectively of the first linear layer. $W2$ and biases are weights and biases of the second linear layer.

The first sub-layer in the decoder is a masked multi-head attention layer. The masking mechanism is used to mask the future words. The second sub-layer in the decoder is the encoder-decoder multi-head attention where the query comes from the previous decoder layer and key and value parameters come from the encoder layer. The output is then passed through the FFN as per (3.9) followed by a linear layer and softmax to obtain the output probabilities. Residual connections are used across all sub-layers in the encoder and decoder. After every sub-layer, a layer normalization is implemented where the inputs are normalized across the features.

3.2.5 Sequence-to-sequence modeling with Reinforcement Learning

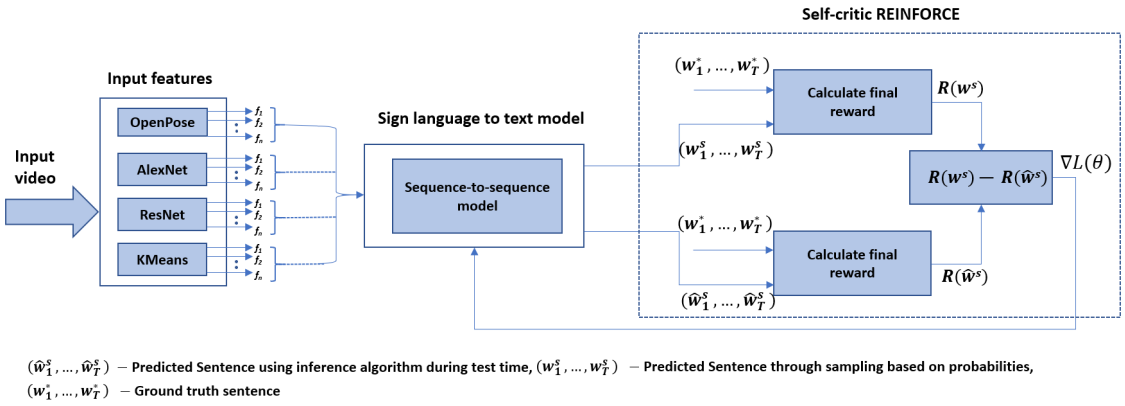


Figure 3.4: Sign language to text model using reinforcement learning.

Inspired by [61, 60] we investigate the performance of a sign language to text model using RL to update the weights and contrast and compare its behavior with different input feature vectors and different models. RL learns what actions need to be taken to maximize a certain reward. Actions are not initially defined but it is something that the model discovers while obtaining either positive or negative rewards [83]. Most of the sequence-to-sequence methods use "teacher-forcing" while training to lead to exposure bias issues during test-time. RL mitigates this exposure bias problem. Casting our model in the RL archetype, the type of model used (sequence-to-sequence, or transformer) will act as an agent with a policy p_θ. The environment will be a video from a sign language dataset, and the state or observation will be the frames from the video. The action for this task would be predicting the next word. Initially, the reward is set to 0 until EOS token is received. At the end of all the tokens, the final reward is denoted as R. The goal is to minimize the negative expected reward as shown in (3.10). The architecture is as shown in Fig. 3.4.

$$L(\theta) = -\mathbb{E}_{w^s \sim p_\theta}[R(w^s)], \tag{3.10}$$

where $w^s = (w_1^s, ..., w_T^s)$ are the words sampled from the model from time steps $1 - T$. Adapting from [83, 62], we utilize REINFORCE [62] with a baseline which is computed as shown in (3.11).

$$\nabla L(\theta) = -\mathbb{E}_{w^s \sim p_\theta}[R(w^s)\nabla_\theta log p_\theta(w^s)] \tag{3.11}$$

The self-critic REINFORCE algorithm is calculated by using inference algorithm at test time for calculating the reward.

$$\begin{aligned} R(w^s) &= Metric((w_1^s, ..., w_T^s), (w_1^*, ..., w_T^*)) \\ R(\hat{w^s}) &= Metric((\hat{w}_1^s), ..., (\hat{w}_T^s), (w_1^*, ..., w_T^*)) \end{aligned} \tag{3.12}$$

where $(w_1^s, ..., w_T^s)$ is the sentence generated through sampling based probabilities. $((\hat{w}_1^s), ..., (\hat{w}_T^s))$ is the sentence generated using the inference algorithm during test time and $(w_1^*, ..., w_T^*)$ is the ground truth sentence. The $Metric$ based on which the reward in (3.12) is calculated can be any evaluation metric like CIDEr or WER.

The word sampled at time step t, $\hat{w}_t$ can be obtained by taking the max probability following the

greedy decoding technique as shown in (3.13).

$$\hat{w}_t = \arg\max_{w_t} p(w_t) \tag{3.13}$$

The Self-critic loss function can now be written as:

$$\nabla L(\theta) \approx -(R(w^s) - R(\hat{w}))\nabla_\theta log p_\theta(w^s) \tag{3.14}$$

The final part of (3.14), $\nabla_\theta log p_\theta(w^s)$, is effectively equivalent to $\frac{\nabla p_\theta(w^s)}{p_\theta(w^s)}$ which is the column vector of the partial derivatives of $p_\theta(w^s)$ divided by the policy evaluation for that particular w^s. The form in the equation is used specifically to make it possible to update the loss function when doing gradient descent.

In Fig. 3.4, the symbol $\nabla L(\theta)$ is intended to represent the final gradient expression of the RL algorithm. This allows for the generation of a scalar loss value which the sign language to text models can use to backpropagate and learn. This can be contrasted to a more traditional loss function such as cross entropy.

3.3 BiLingual Evaluation Understudy (BLEU)

BLEU [84] score is a popular metric that compares a predicted caption with one or more ground truth captions. The metric is based on a calculation that compares the n-grams (number of words per group) from the predicted caption to n-grams in the ground truth caption. Examples of n-grams are unigrams (comparing single words), bigrams (comparing two words), trigrams (comparing three words), *etc.* The unigram precision is calculated by counting the number of times a word occurs in the predicted sentence, this is then clipped to the maximum number of times that particular word has occurred in the ground truth sentence. This result is divided by the total number of words in the predicted sentence. However, this method of measuring the BLEU score fails whenever the length of the predicted sentence is less than the ground truth sentence. To overcome this problem, the predicted sentence is penalized whenever its length is less than that of the ground truth sentence. This penalty is called Brevity Penalty (BP) [84].

4. Single-feature Sign Language Translation

In this section, we discuss experiments performed on the models discussed in section 3 using various input features and datasets. Below, we discuss in detail the input features used, the training details, and performance analysis.

4.1 Input Features

We extract various features from each frame from the sign language datasets and pass them in as inputs to the different models under consideration. These features have been meticulously chosen to obtain good performance. We extract OpenPose [48] features obtaining body, hand and face joint locations, and AlexNet [85], and ResNet [86] CNN feature from the video. The CNN feature extraction models are pre-trained on ImageNet [87] dataset to extract a multi-dimensional feature vector from the visual frames. We also evaluate the use of KMeans cluster IDs based on OpenPose joint locations. The details on each of the feature extractors are described in the following subsections.

4.1.1 OpenPose Features

Using OpenPose [48] we extract 25 body-joint keypoints, 21 keypoints for each hand, and 70 facial landmark keypoints. x, y and *confidence* are obtained for each of these 137 keypoints. Out of these we only choose the x, y coordinates of the joints. So for each frame, we have an input vector of 274 points. We keep the order of the joints the same as the original authors [48]. Because CSL has full-body information (see Fig. 2.1) we use all 274 points, but for the GSL and ASL datasets, only the upper body landmarks are present. The other landmarks are zeroed out. From Fig. 4.1 it can be seen that there are variations in the dataset based on where the person is standing relative to the camera position. To take care of this variation, We obtain a canonical and normalized representation based on the center body points. After mapping to canonical form, all the body points are centered to the origin and scaled to the same size. Fig. 4.1 demonstrates the value of canonical representations.

We additionally perform frame-to-frame smoothing of the OpenPose points. The OpenPose points are firstly median filtered to remove any noise in the data and secondly temporally smoothed using Savitzky-Golay (SavGol) [88] filtering mechanism.

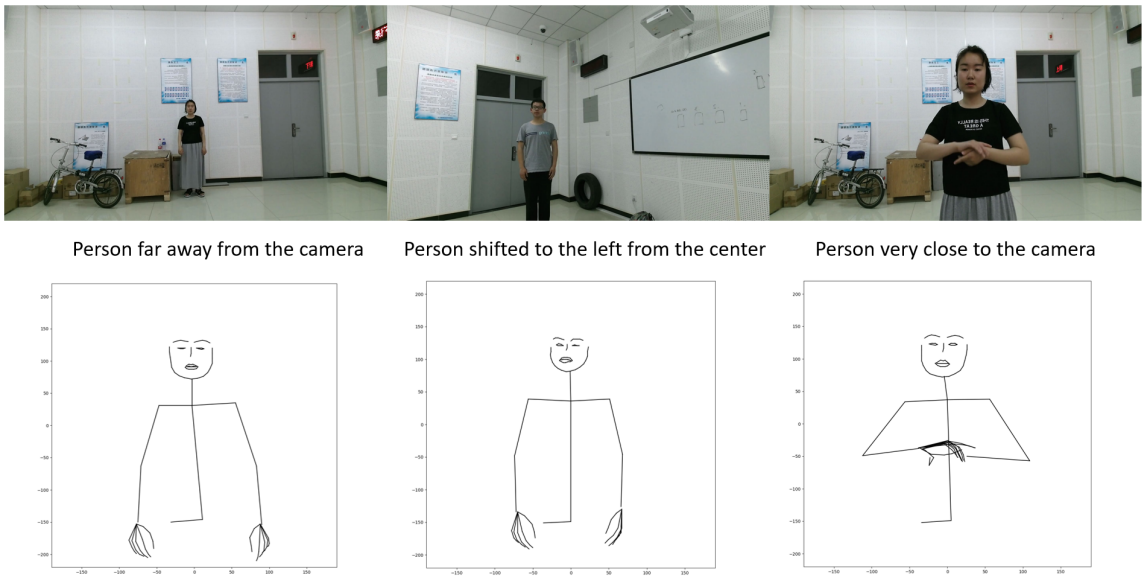


Figure 4.1: Top figure shows sample frames from the Chinese Sign Language Dataset (CSLD), bottom figure shows the respective canonical representations of the frames.

After obtaining the canonical and smoothed points, normalization is performed. We perform two types of normalization's, *Min-Max* and *Standardization*.

Min-Max Normalization: *Min-Max* technique normalizes all the points such that the values fall in the $0-1$ range. *Min-Max* normalization can be calculated as (4.1).

$$X_{\text{normalised}} = \frac{X - \min(X)}{\max(X) - \min(X)} \tag{4.1}$$

Where X is the original OpenPose feature vector, $X_{normalized}$ is the normalized representation of the OpenPose features, $min(X)$ is the least value in the data, and $max(X)$ is the maximum value. The expression $max(X) - min(X)$ gives the range obtained by subtracting the minimum value in the distribution from the maximum value.

Standardization: The keypoints location provided by OpenPose moves around a true value. *Standardization* technique considers the mean and standard deviation to address this issue. Consider two

feature vectors $X = (x_1, x_2, x_3, ..., x_N)$ and $Y = (y_1, y_2, y_3, ..., y_N)$ containing x and y coordinate location of human keypoints respectively. Both the feature vector contains N elements where N in our case in 137, signifying 137 human keypoints identified. The mean and standard deviation is calculated as (4.2).

$$X_{\text{standardised}} = \frac{X - \overline{X}}{\sigma(X)}, Y_{\text{standardised}} = \frac{Y - \overline{Y}}{\sigma(Y)} \tag{4.2}$$

Where $\overline{X}$ and $\overline{Y}$ are mean of feature vector X and Y respectively. σ denotes the standard deviation.

4.1.2 CNN Features

We explore different CNN features to see which visual features stand out. The features include pretrained AlexNet (4096 dim) [85], pretrained ResNet50 (2048 dim) [86], pretrained EfficientNet-B7 (2560 dim) [89], pretrained InceptionV3 (2048 dim) [90]. These visual extractors are pretrained on ImageNet [87]. We also perform end-to-end training using InceptionV1 (1024 dim) [91] and ResNet50 and use the trained network to extract the corresponding features.

4.1.3 Kmeans from OpenPose Features

We developed a pose dictionary where each frame belonged to a cluster based on the similarity of the keypoints and cluster center. Hence the entire video was represented as a vector of cluster IDs. 9521 clusters gave the best performance.

Fig. 4.2 shows how KMeans groups similar poses together. Here we plot some frames belonging to particular cluster IDs.

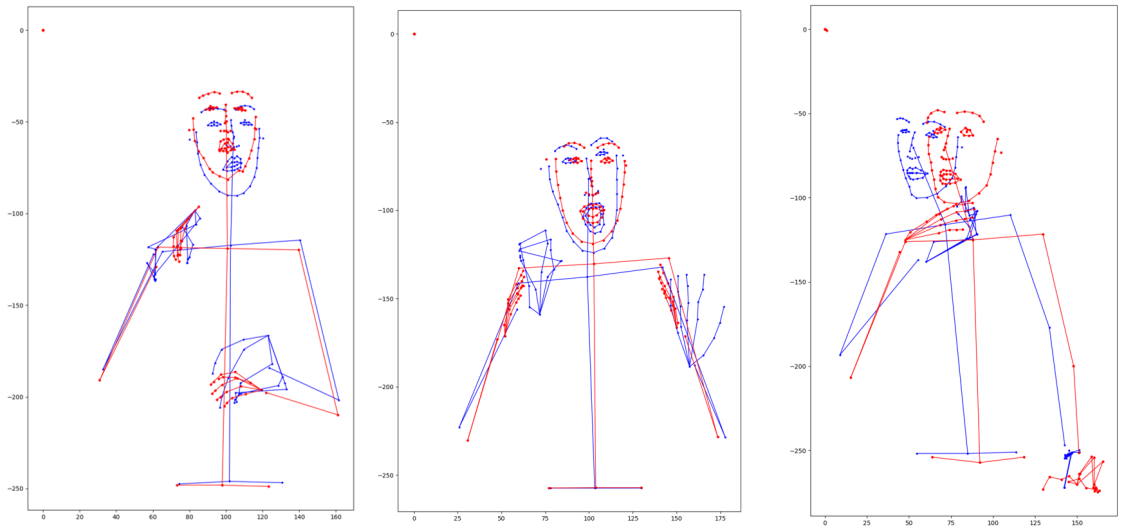


Figure 4.2: KMeans cluster points (red) and original OpenPose points (blue) are plotted for a few random frames from the GSL dataset. The x and y axis represent the body joint locations.

4.2 Training details

This section contrasts different sign language to text models trained and evaluated on the GSL, ASL, and CSL datasets. Different ablations are performed by taking various kinds of feature vectors as input. For GSL, the models are trained on 7096 videos and validated and tested on 519 and 642 videos respectively. For CSL, 2240 videos have been chosen based on sentence repetition. The models are trained on 1790 of these 2240 videos and tested on 450 videos. After splitting the main story videos from the ASL dataset, we obtain 1031 training and 340 test videos. Other training details are mentioned according to the specific model under consideration in the subsections below. Unless otherwise mentioned, for GSL the maximum frame length and maximum caption length are set to 300 and 30, whereas, for CSL it is set to 240 and 20, and for ASL it is set to 225 and 10, respectively. Each batch requires the same input and output length for processing. For this reason, most of our sequence models are designed in a way that can handle a fixed input length and a fixed output length. To avoid a lot of padding, the maximum frame is chosen to be the average frame length and the caption length is chosen to be the average caption length in the respective datasets. For training, we use NVIDIA RTX 2080 Ti graphics processor. Listed below are training details for specific models.

1. Sequence-to-sequence model: The model is trained using Adam optimization with a learning rate and a decay of $1e^{-04}$ and $1e^{-05}$, respectively. We set the dimensionality of the hidden space to 256 for all the datasets the model is trained on. Results are reported on a bi-directional LSTM model with four layers. The model is trained with a batch size of 32. On average, the model takes 14 hours or 150 epochs to train on the GPU.

2. Sequence-to-sequence model with attention: The model is trained using Adam optimization with a learning rate of $1e^{-05}$. Training is performed for $150,000$ steps. We use Luong [22] attention for this sign language translation model. Training takes approximately four days on one GPU. We use a batch size of one as it proved to give the best results in [9]. The model is trained with 1000 dimensional hidden units and four layers of bi-directional LSTM.

3. Transformer Model: While training the transformer model, for each batch the maximum length of the sequence in the batch is chosen as the number of frames and likewise for the number of captions. Batch normalization and softsign [92] activation are used on input features before feeding to the network as it gives us good results. The model is trained for 100 epochs with a batch size of 32 and Adam optimizer.

4. Reinforcement Learning: The model is trained using Adam optimization with a learning rate and a decay of $5e^{-04}$ and $5e^{-05}$, respectively. We set the dimensionality of the hidden space to 256. Results are reported on a bi-directional LSTM model with two layers. The model is trained with a batch size of 32. On average, the model takes 15 hours or 280 epochs to train on the GPU. The model is pre-trained using cross-entropy for 56 epochs and then trained using self-critical sequence training for the remaining 224 epochs.

4.3 Performance Analysis

We begin our analysis by considering the GSL dataset mainly due to two reasons, one, the dataset consists of samples portraying a controlled environment, and two, because the dataset has the highest training samples and a good distribution of samples in the validation and test sets as compared to the other datasets.

4.3.1 Variants of OpenPose for sign language translation

OpenPose joints keypoints gained popularity from gesture and action recognition tasks [46, 47, 48, 49] as already see in Chapter 3. We want to explore its performance for sign language translation. In addition, OpenPose joints are very human interpretable. It highlights the importance of joints for particular signs. We want to investigate how different OpenPose joints contribute to the performance of sign language translation. The ablations with OpenPose features on the GSL dataset are shown in Table 4.1.

Observations:

From our observation, hands are the most important keypoints from OpenPose that are useful for translation. Even though hands alone does not yield high scores, it improves when combined with face and body, whereas it jumps significantly when all three are combined.

Table 4.1: OpenPose ablation results on sequence-to-sequence model without attention on the GSL dataset.

OpenPose Features	Set	BLEU 1	BLEU 2	BLEU 3	BLEU 4
Hands	Validation	7.31	1.36	0.34	0.07
	Test	12.14	1.71	0.55	0.12
Body	Validation	4.63	0.88	0.18	0.01
	Test	6.29	0.82	0.14	0.01
Face	Validation	2.54	0.46	0.04	0.01
	Test	2.23	0.40	0.04	0.01
Hands + Body	Validation	18.11	6.50	4.42	4.06
	Test	20.68	7.01	4.65	4.33
Hands + Face	Validation	18.13	5.92	4.51	4.06
	Test	17.68	5.76	4.47	4.04
Hands + Body + Face	Validation	**23.31**	**8.73**	**6.49**	**5.68**
	Test	**23.49**	**8.52**	**6.36**	**5.55**

4.3.2 Choosing between Vanilla RNN, GRU, and LSTM

We perform sign language to text translation on the GSL dataset using Vanilla RNN, GRU, and LSTM bi-directional models by feeding OpenPose key points as input. From the results shown in Table 4.2 we can see that Vanilla RNN performs the worst when compared to LSTM and GRU.

Observations: We can thus infer that LSTM and GRU are capable of handling long term dependencies when compared to Vanilla RNN. For all the sequence-to-sequence modeling used in our experiments henceforth, we will be using LSTM as the RNN model.

Table 4.2: Ablations on basic sequence-to-sequence models without attention on the GSL dataset using OpenPose points without frame-to-frame smoothing.

Model	Set	BLEU 1	BLEU 2	BLEU 3	BLEU 4
Vanilla RNN	Validation	12.67	4.09	3.17	2.79
	Test	14.31	4.68	3.43	2.92
GRU	Validation	20.16	7.41	5.29	4.66
	Test	20.40	7.56	5.63	4.98
LSTM	Validation	**23.62**	**8.89**	**6.54**	**5.71**
	Test	**22.89**	**8.43**	**6.20**	**5.57**

4.3.3 Ablations with different CNN features

To choose an appropriate CNN architecture, we perform ablations using the embedding features extracted from the architectures. These CNNs (ResNet50, AlexNet, EfficientNet-B7, InceptionV1, V3) are pretrained on ImageNet and used as feature extractors. Whereas, we performed end-to-end training by freezing the first three layers of InceptionV1 and ResNet50 in a sequence-to-sequence setup. As the end-to-end training needs a lot of memory and takes many days to converge, our models did not have the opportunity to learn as well. All the features were evaluated using the transformer model. The results of the ablations are presented in Table 4.3.

Observations: As seen in Table 4.3, pretrained ResNet50 features provide the best results. Going forward for all the ablations with CNN features we will be using ResNet50 extracted features.

From our analysis, we have observed that SLT datasets benefit from features extracted using deep networks. When comparing with shallow networks (AlexNet), deep networks (ResNet50) provides better features giving better performance in SLT. Empirically, from our studies, we have seen that wider networks do not perform as well. So Inception v1, Inception v3, EfficientNet-B7 (even though is not only wide but also deep) do not seem to perform as well as a purely deep network. The deep network architecture and skip-connections in ResNet50 prove to be the most beneficial for SLT yielding the best results.

Table 4.3: Ablation results on different CNN features using the transformer model on the GSL dataset

CNN Features	Set	BLEU 1	BLEU 2	BLEU 3	BLEU 4
Pretrained ResNet50	Validation	**24.75**	**16.04**	**11.56**	**8.95**
	Test	**23.67**	**14.58**	**10.29**	**8.00**
Pretrained AlexNet	Validation	23.81	14.71	10.65	8.32
	Test	23.00	13.86	9.75	7.5
Pretrained EfficienetNet-B7	Validation	20.84	12.4	8.78	6.9
	Test	18.84	11.36	8.06	6.31
Pretrained Inceptionv3	Validation	19.90	11.75	8.44	6.67
	Test	18.71	11.08	7.93	6.19
Pretrained Inceptionv1	Validation	17.7	10.26	7.28	5.72
	Test	18.18	10.63	7.39	5.67

4.3.4 Experiments and ablations on GSL dataset

From the experiments and ablations in Table 4.4, we study how different feature inputs contribute towards improving the model and how attention plays an important role in sign language to text translation. Visual features from ResNet50 and location-based features from OpenPose smoothed version perform better than other features. Between the sequence-to-sequence models, attention mechanism predicts *bi-grams* and longer *n-grams* better than the sequence-to-sequence model without attention. The transformer model performs further better than the sequence-to-sequence model with attention in predicting *bi-grams* and more due to its extensive multi-head attention and self-attention mechanism. The results of the ablations are presented in Table 4.4.

Observations: Compared to the other models evaluated the transformer can learn more weights and parameters which contribute towards its good performance on the GSL dataset as shown in Table 4.4. The multi-head self-attention mechanism in the transformer model can learn complex representations for every frame. The self-attention in the transformer model helps to learn inter-dependencies between different frames depending upon the position of the frames and the ground truth caption. The self-attentions are calculated in parallel, eight times, referred to as eight heads and concatenated together forming multi-head attention. This parallelization has an advantage in the transformer model as against the sequential sequence-to-sequence models. The advanced architecture of the transformer model helps in learning long-term dependencies when compared to the

Table 4.4: Ablation results on different feature inputs using the GSL dataset.
NS - No Smoothing, WS - With Smoothing,
s2s - basic sequence-to-sequence model without attention,
s2s with att - sequence-to-sequence model with attention.

Input Features	Model	Set	BLEU 1	BLEU 2	BLEU 3	BLEU 4
OpenPose (NS)	s2s	Validation	23.62	8.89	6.54	5.71
		Test	**22.89**	8.43	6.20	5.57
	s2s with att	Validation	17.64	11.42	8.52	6.78
		Test	18.50	11.92	8.73	6.95
	Transformer model	Validation	**24.2**	**15.28**	**11.08**	**8.69**
		Test	22.69	**14.61**	**10.38**	**8.09**
OpenPose (WS)	s2s	Validation	23.31	8.73	6.49	5.68
		Test	23.49	8.52	6.36	5.55
	s2s with att	Validation	20.65	13.39	9.80	7.68
		Test	19.65	12.36	8.97	7.00
	Transformer model	Validation	**24.82**	**15.87**	**11.53**	**9.07**
		Test	**23.52**	**15.24**	**11.00**	**8.67**
ResNet50	s2s	Validation	21.55	7.97	5.76	4.96
		Test	20.90	7.81	5.08	4.22
	s2s with att	Validation	23.01	14.48	10.20	7.79
		Test	21.71	13.89	10.07	7.87
	Transformer model	Validation	**24.75**	**16.04**	**11.56**	**8.95**
		Test	**23.67**	**14.58**	**10.29**	**8.00**
KMeans	s2s	Validation	**15.68**	3.67	3.12	2.74
		Test	**16.93**	3.90	3.32	2.95
	s2s with att	Validation	12.75	6.92	5.06	4.07
		Test	12.78	7.30	5.51	**4.54**
	Transformer model	Validation	14.02	**7.47**	**5.44**	**4.41**
		Test	13.65	**7.46**	**5.57**	**4.54**

sequence-to-sequence models. Although ResNet50 and OpenPose input features perform very similar with the highest BLEU scores on the transformer model, OpenPose features are less expensive to get than ResNet50 features.

4.3.5 Experiments and ablations on CSL dataset

Preliminary results on the CSL dataset [3] are shown in the Table 4.5. The CSL dataset chosen for training is a very small subset of the whole dataset. Due to the language complexity and abnormalities in the dataset, there is still room for improvement in the Chinese Sign Language recognition models. BLEU 1 scores are considerably high when compared to BLEU 2 scores because a few characters like "我 (I)" occur 2103 times in the dataset followed by "是 (is)", "他 (he)", "你 (you)", repeating around 1000 times. For the CSL dataset, OpenPose feature inputs work better than CNN

features because of the structure of the dataset. As explained in Section 4.1.1 and Fig. 4.1 the OpenPose canonical form obtain a scaled and center version of points for all the frames, whereas the CNN features still take in as input the original frames where the person signing may not be at the center and the same distance from the camera. For such a real-life scenario dataset consisting of different abnormalities and noise, OpenPose proves to be beneficial while performing sign language to text translation.

The attention mechanisms (sequence-to-sequence with attention and transformer model) perform very well with OpenPose input features. The transformer model, however, performs the best with high BLEU 1 to BLEU 4 scores for the pose based inputs. Whereas, the CNN feature input overall does not do very well due to abnormalities in the dataset.

Observations: In situations where the CNN features are not as good due to the structure of the dataset, OpenPose feature inputs help in elevating the results.

Table 4.5: Ablation results on different feature inputs using the CSL dataset.
NS - No Smoothing, WS - With Smoothing,
s2s - basic sequence-to-sequence model without attention
s2s with att - sequence-to-sequence model with attention.

Input Features	Model	Set	BLEU 1	BLEU 2	BLEU 3	BLEU 4
OpenPose (NS)	s2s	Test	14.36	1.60	0.04	0.01
	s2s with att	Test	39.45	34.44	32.55	31.43
	Transformer model	Test	**52.04**	**48.94**	**47.86**	**47.25**
OpenPose (WS)	s2s	Test	19.68	2.32	0.12	0.01
	s2s with att	Test	36.13	31.11	29.48	28.63
	Transformer model	Test	**43.03**	**38.93**	**37.52**	**36.76**
ResNet50	s2s	Test	**17.41**	1.19	0.12	0.01
	s2s with att	Test	14.01	5.16	3.55	2.93
	Transformer model	Test	13.2	**6.04**	**4.58**	**4.02**

4.3.6 Experiments and ablations on ASL dataset

Preliminary results with the ASL dataset are shown in Table 4.6. The Sequence-to-sequence model with attention follows a similar trend like GSL and CSL in predicting longer *n-grams* better for different feature inputs. It is evident from the Table 4.4, 4.5, and 4.6 that the organization of the dataset highly influences the results. While ASL is very similar to the GSL dataset, it has fewer samples and covers a widespread topic rather than just weather information (like GSL). It is very

different from the CSL dataset with respect to the number of signers, the number of samples, and the type of samples. The OpenPose and CNN feature inputs perform comparatively the same for the ASL dataset using the attention models. The sequence-to-sequence model with attention performs the best when compared with the transformer model and the sequence-to-sequence model without attention. The transformer model does not provide a boost in BLEU scores as seen in the other datasets.

Observations: The organization of the ASL dataset is somewhere between the GSL and CSL datasets. Because of the input frames having a controlled background, it performs better than CSL with CNN features but due to the poor quality of frames when compared to CSL, it incurs a performance hit when using OpenPose points. The poor performance of the transformer model may be due to the ASL being a smaller dataset with fewer repetitions as seen in Fig. 2.2 and with 10% of sentences in the test set seen during training. The CSL dataset is chosen such that the dataset has sentences that are repeated due to this there is a high overlap of the captions between test and train but these are all signed by different signers. GSL, however, being a huge dataset, performs fairly better than ASL even with less number of test samples seen during training. To further understand the ASL dataset, we perform experiments to see how different signers interpret the same sign which is further explained in Section 4.3.8.

Table 4.6: Ablation results on different feature inputs using the ASL dataset.
NS - No Smoothing, WS - With Smoothing,
s2s - basic sequence-to-sequence model without attention,
s2s with att - sequence-to-sequence model with attention.

Input Features	Model	Set	BLEU 1	BLEU 2	BLEU 3	BLEU 4
OpenPose (NS)	s2s	Test	14.68	3.81	2.76	2.31
	s2s with att	Test	**19.83**	**8.45**	**4.79**	2.93
	Transformer model	Test	13.58	6.7	4.35	**3.17**
OpenPose (WS)	s2s	Test	16.85	4.17	3.29	3.02
	s2s with att	Test	**20.20**	**8.62**	**5.21**	**3.43**
	Transformer model	Test	13.57	6.83	4.46	3.34
ResNet50	s2s	Test	**21.66**	4.95	4.43	**4.26**
	s2s with att	Test	19.55	**10.02**	**6.2**	4.25
	Transformer model	test	12.78	5.7	3.38	2.43

4.3.7 Addition experiments and analysis using Reinforcement Learning (RL)

RL is also used as one of the models to analyze how it tackles the *exposure bias* problem when compared to other methods. Using a rewards mechanism, the final gradient is calculated which is then backpropagated to the model under consideration. We report preliminary results with RL on the sequence-to-sequence model without attention using ResNet features on the GSL dataset. The results are shown in Table 4.7.

Observations: The RL model performs better than the sequence-to-sequence model without attention. Thus the RL model without teacher-forcing can perform well when compared to the model which incorporates teacher-forcing thus reducing the *exposure-bias* problem. This model is also able to predict *one-gram* words better than the sequence-to-sequence model with attention. In the future, it would be interesting to see how the RL model performs if attention is introduced.

Table 4.7: Results on the GSL dataset using RL based sequence-to-sequence (s2s) model without attention with ResNet input features.

Model	Set	BLEU 1	BLEU 2	BLEU 3	BLEU 4
s2s without RL	Validation	21.55	7.97	5.76	4.96
	Test	20.90	7.81	5.08	4.22
s2s with RL	Validation	26.97	8.94	7.08	6.48
	Test	25.70	7.87	6.31	5.64

4.3.8 Human as an oracle

We performed an experiment with a human as an oracle (an expert signer) to compare how a human is able to predict the captions on a sign language video versus OpenPose joints visualization for each frame rendered as a video. The OpenPose joints are mainly used for this experiment because the OpenPose annotated body, hands, and face joints are highly interpretable by humans. The results on 340 ASL videos from the test set and OpenPose videos are shown in Table 4.8.

Table 4.8: Human as an oracle experiment on ASL dataset for original (sign language RGB video) and OpenPose generated videos.

Video type	BLEU 1	BLEU 2	BLEU 3	BLEU 4
Original videos	24.86	12.17	6.87	3.81
OpenPose videos	6.32	1.790	0.59	0.14

The low BLEU 4 scores in Table 4.8 for original videos may be attributed to the fact that the ground

truth captions were taken out of context from a storytelling scenario. Therefore, the signs and captions may not necessarily match up. Ideally, 50% BLEU 4 score from human evaluation would indicate that the ground truth is collected efficiently. To test the efficacy of the dataset collection further, we asked another ASL signer to annotate 5 videos and compared the results of the two signers with the ground truth.

Table 4.9: Comparison between ground truth and predicted captions between two human annotators. The yellow cells show agreement while green shows disagreement (Best viewed in color).

No.	Ground truth caption	ASL Signer 1	ASL Signer 2
1.	He called and asked if they had the right kind of dog, and the answer was yes!	You can have	they shouted "you have it?", "yes".
2.	He keeps drinking coffee	Still making out	coffee continued still.
3.	It must be a head trip or something	I was tripping	I'm strict/hardheaded.
4.	As I was driving, I was really bored. So I started thinking about what I could do.	The car ride was boring	Driving's boring.
5.	Did the teacher buy a house yesterday?	Did the teacher buy a house yesterday?	Did the teacher buy a house yesterday?

Observations: From Table 4.9 we can see that the predicted sentences have non-matching *grams* thus leading to low BLEU scores in Table 4.8.

4.4 Conclusion

There is still much to be done to facilitate smooth communication between hearing and non-hearing communities. In this chapter, we have discussed how methodologies evolved from text-to-text translation to sign-to-text translation. We selected four models to evaluate: the sequence-to-sequence model, the sequence-to-sequence model with attention, the transformer model, and the reinforcement-learning based model; to better understand how well they perform for sign language translation. We also explored different input features such as the joints of the body and hands, along with facial landmarks all using OpenPose, CNN features, and KMeans cluster IDs for each frame. To

better understand which sets and subsets of features aid best in improving performance, we performed ablation studies using different combinations of body, hands, and face keypoints as inputs to our models. In addition to ablations with different input features, we also examined different sign language datasets to better understand how the performance varied, based on the complexity (controlled collection and constrained vocabulary) of the sign language videos. We evaluated our models on different sign languages, ASL, GSL, and CSL. Our transformer model outperformed all the other sequence-to-sequence models on the GSL and CSL datasets using OpenPose features. In some cases, ResNet features provided similar results as OpenPose but as seen it is highly dataset dependent. ASL dataset could not leverage the attention mechanisms involved in the transformer model due to being a smaller and noisy dataset. In general, the attention mechanisms involved in the transformer model learned more weights than the sequence-to-sequence models with and without attention, thus leading to significantly better learning capability.

To provide the reader with a comprehensive study for sign language translation, we performed a preliminary experiment using ResNet features with the sequence-to-sequence model without attention and RL. This RL-based model helped in mitigating the exposure bias issue usually seen due to teacher-forcing when sequence-to-sequence models are used. Our RL based sequence-to-sequence model yielded approximately a 1.5-5 point increase in BLEU 1 - BLEU 4 scores. Our human-as-an-oracle experiment completed our extensive study by looking at how a human expert signer would interpret the ASL videos and their corresponding OpenPose-based stick videos. We used OpenPose for comparison as it is visually more understandable for predicting signs when compared to other features.

In conclusion, this section has provided a comprehensive overview of the representations and models used for continuous sign language translation without gloss. We have shown that the transformer model performed the best on the controlled and constrained GSL dataset, especially when combined with either OpenPose or ResNet50 as input features. But for the other less consistent datasets, the results were more varied. Also, the human experiment demonstrated that our ASL dataset is very noisy (with several incorrect captions), resulting in low BLEU scores being obtained even by a human expert and low agreement between two different signers. In the future, we plan to collect a larger and more consistent ASL dataset, to make it more on par with the GSL and CSL datasets.

5. Multi-feature Fusion for Sign Language Translation

5.1 Motivation

The motivation for our current chapter comes mainly from two points:

1. From our previous chapter we can conclude that when good quality frames are presented to the model, some of the features like CNN features provide better information than the others, whereas when the quality of frames is not up to the mark, other features like pose-based features help in contributing to the performance. In a real-life scenario, it is difficult to predict what type of frames/ videos are encountered. The realistic setting could include a lot of clutter in the background, varied illuminations, poor RGB quality, etc,. It is therefore desirable to adapt dynamically based on the quality/ type of input seen.

2. In SL research, the majority of the work has been done using the GSL dataset. GSL is a non-realistic dataset due to it being collected in a controlled environment as seen in Chapter 2. This brings our focus to ASL which is currently a low resource language. As many as 5% of Americans are currently DHH [93] and as such, American Sign Language (ASL) as their primary mode of communication. But in the hearing-centric world we live in, the majority of the general population do not understand ASL and inadvertently require DHH individuals to communicate in ways other than their natural mode of communication. But in spite of its popularity, ASL has received very little computational research attention, probably due to limited data and the complexity associated with being a visual language in a mostly hearing-centric environment.

But the growing successes of translating between spoken languages have inspired machine translations of visual languages such as sign language into spoken/written ones and we discuss several of

these methods in this chapter. But the absence of annotated large-scale, parallel phrase-level ASL datasets and applicable NLP tools potentially qualifies it as a low-resource language.

Hence, in this chapter, we focus towards growing body of work in continuous sign language analysis through the following:

- We collect an ASL dataset by recording expert signers as they sign written English phrases provided. Although relatively small, to the best of our knowledge it is the largest phrase-level ASL dataset available. This ASL linguistic dataset (which we refer to as ASLing) will be made publicly available, to add to the growing body of sign language resources. Details are provided in Section 2.

- We develop two novel dynamic multi-feature fusion architectures, tri-feature late fusion (TFLF) and cross-feature dual fusion (CFDF), attention networks. These networks were designed to translate signing videos of varying quality, focusing purely on translation without gloss, the intermediary written symbolic representation of a sign language. We perform initial tests on the German sign language (GSL) benchmark dataset, RWTH-PHOENIX-Weather, and obtain state-of-the-art (SOTA) translation metrics. We then apply the architectures on ASLing and test for the efficacy of transfer learning from the annotated GSL corpus to the low-resource ASL one.

- We perform experiments using the dynamic multi-feature architectures on ASLing and compare it with single features.

- We explore the explainability efficacy of the models and determine how each of the input features contributes to the final translation. We accomplish this by visualizing the attention weights over time, highlighting the influence of each feature over the sequence.

- We improve the performance of the unconstrained and uncontrolled ASLing dataset by transfer learning from the larger well-tested German sign language (GSL) dataset.

In a real-life scenario, the input sign language video cannot always be captured in a controlled, and well-balanced environment. Hence, in this work, we perform automated SLT on an unconstrained and uncontrolled sign language dataset (ASLing) using a multi-feature fusion architecture shown in

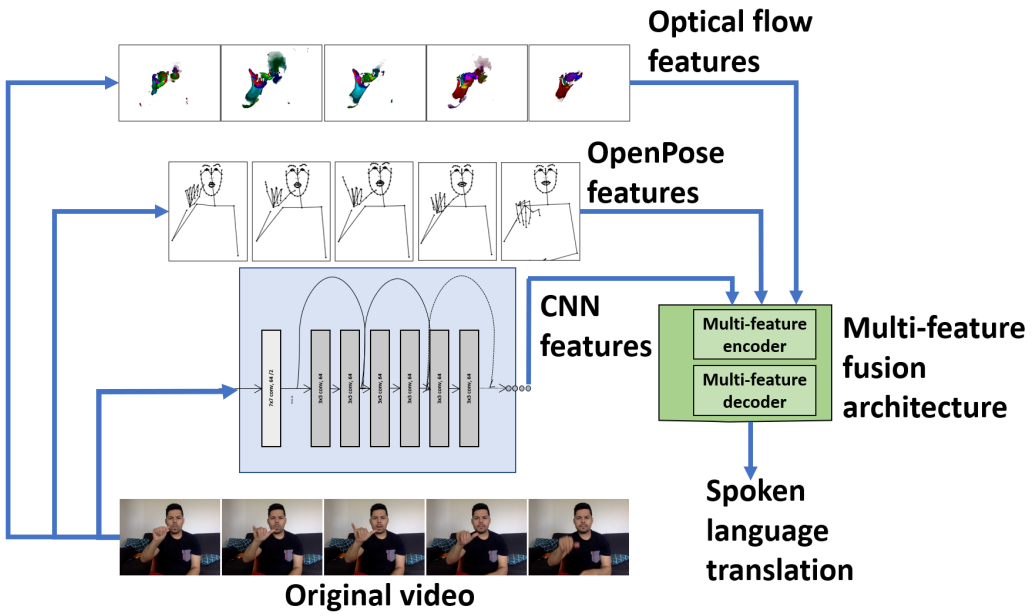


Figure 5.1: Multi-feature fusion for sign language translation.

Fig. 6.1. The model is adaptive to the variations in the input frames and attends to different features based on the level of information rendered.

5.2 Related Works

Some popular related tasks target different modalities such as vision, audio, language, and others target different features extracted from these modalities. *Zadeh et al.,* [94] used a system of LSTM with a multimodal state memory to store the dependencies like [95], but used a dynamic fusion graph (DFG) to dynamically attend to three modalities - vision, language, and audio for sentiment analysis. *Tsai et al.,* [96] also considered the same modalities, audio, vision, and language but used the transformer model to learn the importance between the different modalities with a goal of classification. *Sahay et al.,* [97] built upon [96] by passing a fused signal to the cross-modality transformer model. This fused signal was obtained by performing a low-rank fusion of the input features. Pre-fusion and post-fusion techniques were explored for video captioning using different spatial, motion, dense features from the input frames by *Jiang* [98] using RNNs. *Li et al.,* [99] used the text, video, and audio modalities for scene description by concatenating different modality inputs forming a long sequence of inputs fed to a transformer model thus attending. One of the recent work

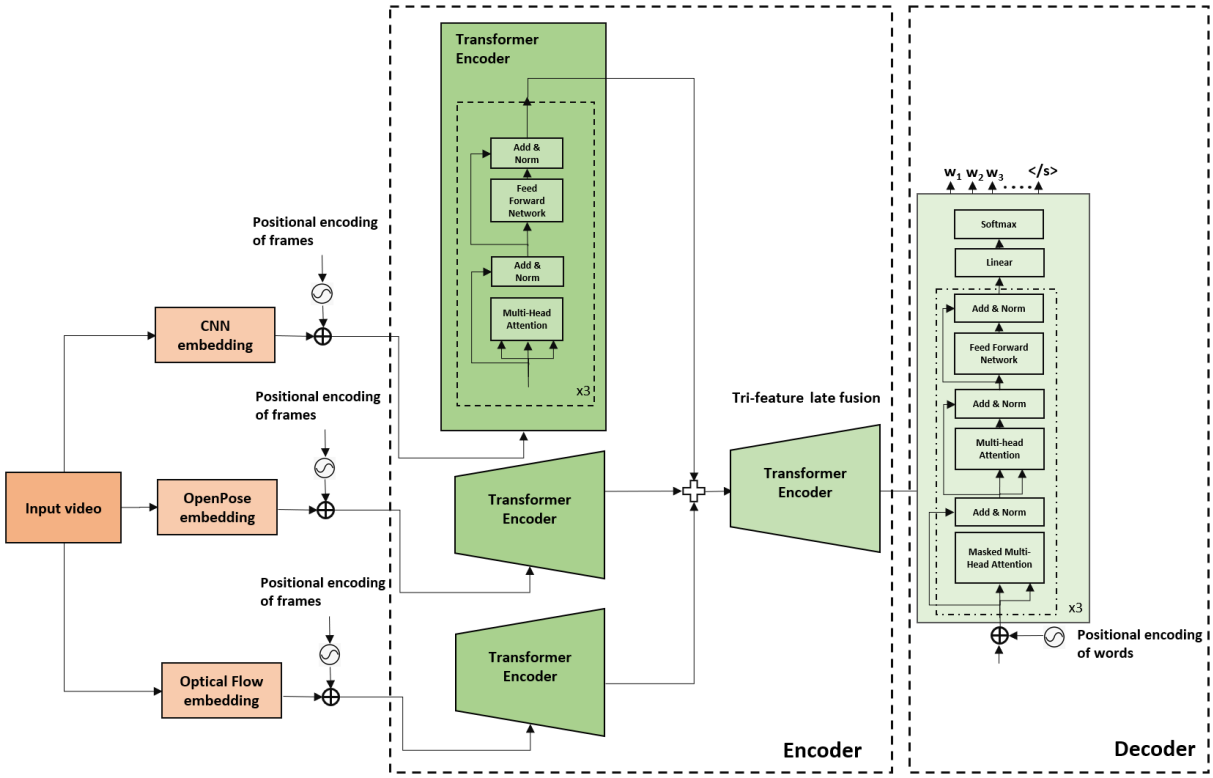


Figure 5.2: Tri-feature late fusion (TFLF) model for sign language translation (best viewed in color). For space conservation only one transformer encoder block is expanded.

done by *Huang et al.,* [100] consider audio and video for emotion recognition task perform model level fusion by independently learning the attention in the audio and video modalities and fusing them as input to another transformer model thus learning cross-modality attention. *Sulubacak et al.,* [101] provides a comprehensive study of different applications, modalities, and architectures for multimodal machine translation tasks.

Sign language translation to spoken language is a challenging task because it is so dependent on face and body movements as well as visual, spatial-temporal, location-based features derived from the input video. *Papadimitriou et al.,* [102] explores multiple features on three different datasets but mostly targets word-level or gloss-level sign language recognition.

5.3 Dynamic multi-feature for American Sign Language Translation

5.3.1 Input Features

To take full advantage of the multi-feature fusion models, we consider three different features from the input video that highlight different modalities. We extract 2048 dimension CNN ResNet50 [86] features for each frame, pretrained on ImageNet [87]; these provide information on the visual RGB frames. We obtain 25 points for the body, 21 points for each hand, and 70 face points from OpenPose [48]. These points are (x, y) locations of the body, hands, and face. We convert these raw locations into a canonical, smoothed, and normalized form which helps the model in better learning and training without exploding gradients. The canonical form is obtained by scaling and centering the points. To perform smoothing we use Savitzky-Golay (SavGol) [88] and perform frame-to-frame smoothing and finally normalize the points to fall between $0 - 1$. We also extract 2048 dimension vector for each frame from optical flow [103, 104] which provide the flow-based information between consecutive frames.

Optical Flow:

The optical flow information for SLT captures the signs and movements of the signer from one frame to the next until the end of the video. Typically, optical flow is categorized as sparse optical flow and dense optical flow. Sparse optical flow helps in identifying edges and corners, thus considering only some of the pixels in the frame whereas dense optical flow considers all the pixels in the frame and thus helps in capturing movements in the frames. To calculate optical flow we use TV-L1 [104] algorithm. This algorithm minimizes a function using L1 norm and regularization by using the total flow variation. TVL1 algorithm is better at handling frames of poor quality and provides better optical flow information than some of the traditional algorithms [105, 106] as seen in Figure. 5.3. The algorithm initially calculates small displacements. Images are downsized to detect displacements that are bigger than one pixel, and later are upsampled to the original scale. Usually, the first step in the TVL1 algorithm is to form a pyramid of scales with downsampled input images. The image is downscaled by convolving with a Gaussian Kernel and bicubic interpolation is used for sampling. The results from the coarsest level are used as the initial point in the finer levels. From the pyramid structure, the optical flow is calculated as different scales.

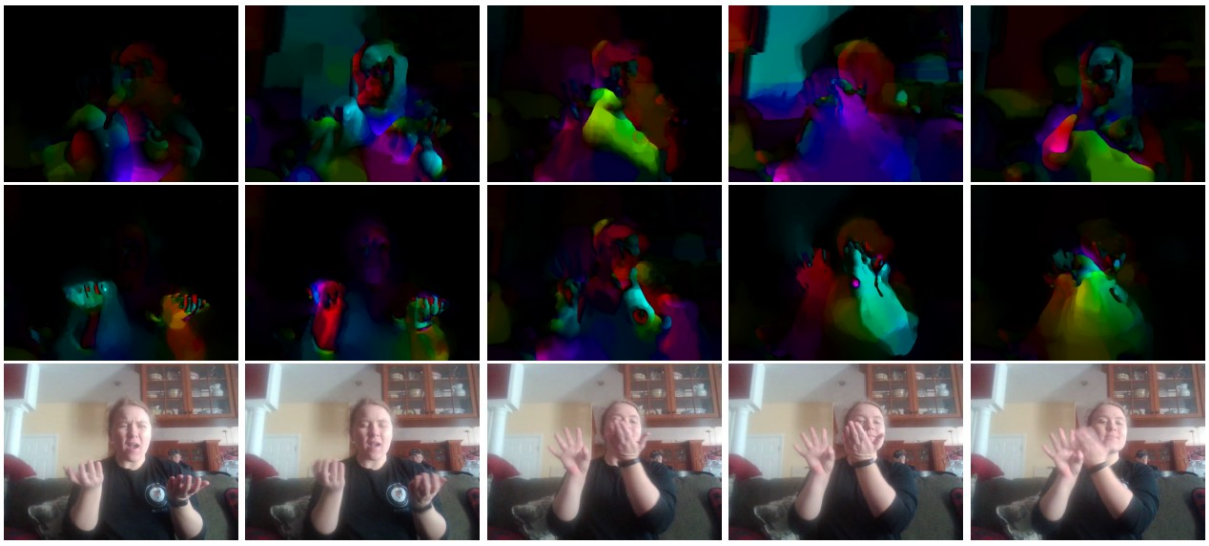


Figure 5.3: Optical Flow output using Franeback Alogrithm (Top); TVL1 Algorithm (Middle). Original frames are displayed on the last row.

We compute Optical Flow using the OpenCV library for TVL1 algorithm [107]. Two consecutive frames are passed as input to calculate the flow movements between them. The flow information for two consecutive frames is dumped into the file as an image. We continue this procedure for each video until the end of the frame is encountered. We pad the videos as needed. These images are then passed through pre-trained ResNet50 [86] to extract a 2048 size feature vector for each frame.

5.3.2 Tri-feature late fusion

The TFLF transformer model architecture is shown in Fig. 5.8. Three different input features (CNN, OpenPose (OP), optical flow(OF)) extracted from the sign language video, highlighting different modalities are fed to standalone transformer encoders. For each of the feature embeddings, the positional encoding is added to consider the frame order as shown in (5.1).

$$\begin{aligned} PE_{(pos,2i)} &= sin(pos/10000^{2i/d_{model}}) \\ PE_{(pos,2i+1)} &= cos(pos/10000^{2i/d_{model}}) \end{aligned} \tag{5.1}$$

This input is then fed to a 1D convolutional layer and the embeddings are fed to the transformer encoder which learns multi-head self-attention. The query (Q), key (K), and values V for each of

the individual transformer encoders are calculated as (5.2).

$$\begin{aligned} K_f &= linear\_key(input_f) \\ Q_f &= linear\_query(input_f) \\ V_f &= linear\_value(input_f) \end{aligned} \tag{5.2}$$

where, *linear_key*, *linear_query*, and *linear_value* are three individual linear layers learnt for the input features (input_f), where, $f \in \{CNN, OP, OF\}$.

For all the features, the multi-head self-attention layers learn and attend to different aspects within the individual features contributing towards improving the performance for the sign language translation task. The attention mechanism for one of the transformer encoder layer is shown in (5.3).

$$Attention(K_f, Q_f, V_f) = softmax(\frac{Q_f K_f^T}{\sqrt{d_k}})V_f \tag{5.3}$$

The feed-forward network is a linear layer calculated as shown in (5.4).

$$FFN(x_f) = max(0, x_f W1 + b1)W2 + b2 \tag{5.4}$$

$W1$ and $b1$ are the weights and biases respectively of the first linear layer. $W2$ and biases are weights and biases of the second linear layer. x_f is the output from the previous block. Only one encoder layer is shown for illustration purposes, but during implementation, 3 layers are used. The output from each of these encoder layers is then fused and fed as input to the TFLF block that attends to the information collectively from all the three sets features under consideration. In this way, late fusion is performed. Consider y_f the output from each of the add and normalization blocks of the transformer encoder, the TFLF input can thus be written as (5.5).

$$input_{tflf} = [y_{f_1} : y_{f_2} : y_{f_3}] \tag{5.5}$$

y_{f1}, y_{f2}, and y_{f3} are the outputs from the respective transformer encoders processing CNN, OP, and OF features respectively.

The TFLF transformer encoder equations are similar to 5.1 - 5.5). The output of the TFLF transformer encoder is fed to the decoder. We incorporate 3 decoder layers in our work. The decoder takes in as input the word embeddings and performs masked-multi head attention by masking the future words. TFLF encoder output is fed to the multi-head attention block in the decoder where it learns the encoder-decoder attention and predicts the words after passing through a feed-forward network, linear, and softmax layers.

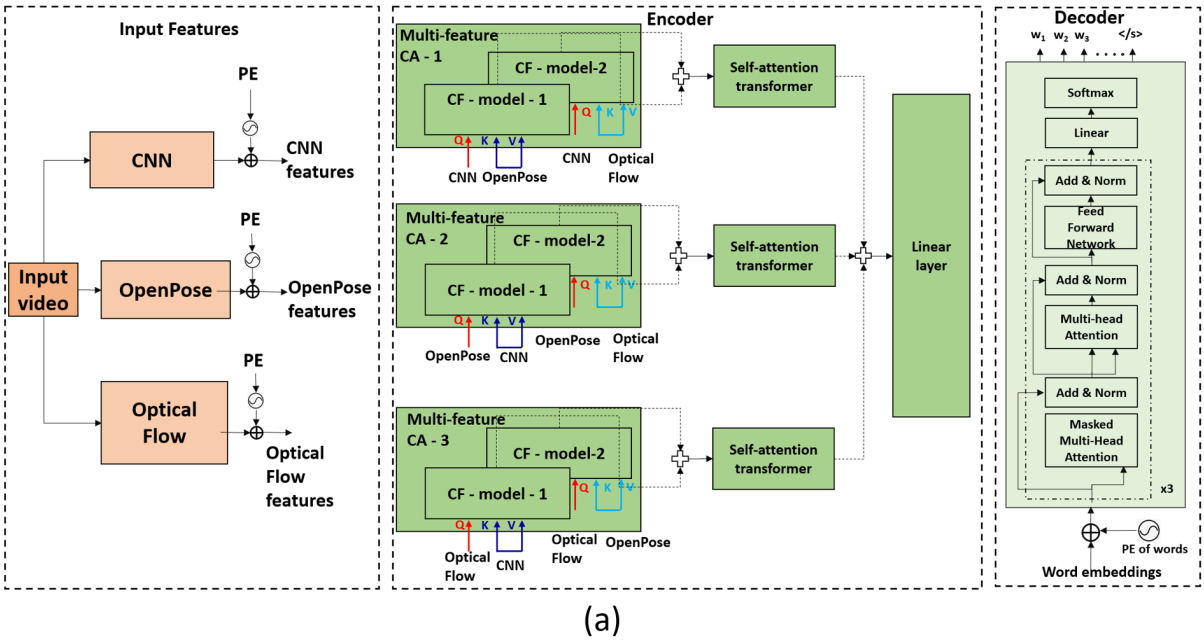


Figure 5.4: Cross-feature dual fusion (CFDF) model for sign language translation.

5.3.3 Cross-feature dual fusion

The cross-feature dual fusion (CFDF) architecture is shown in Fig. 5.4. Three different input features are simultaneously fed to the multi-feature cross-attention (CA) block after adding the positional encoding (PE) information. These inputs are passed through a 1D convolutional network before passing to the next stage.

To understand the CFDF architecture, let us consider one cross-attention block (Multi-feature cross-attention-1) from Fig 5.4, details expanded and shown in Fig. 5.5.

Equation (5.6) describes how the attention for the two cross-feature transformer models attending on the CNN features as the base feature is calculated:

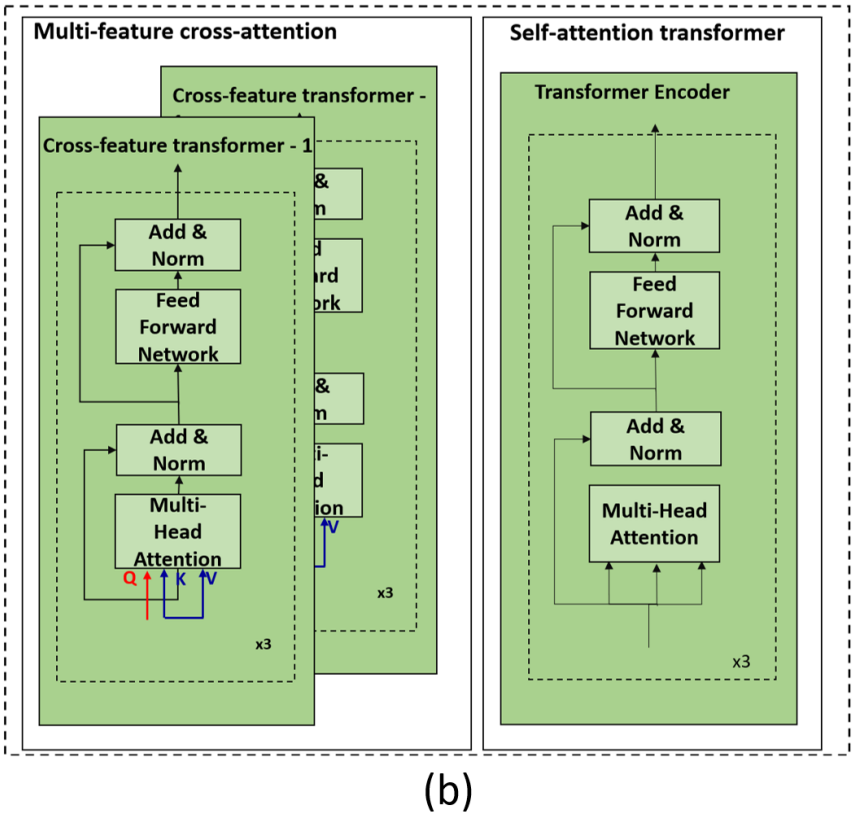


Figure 5.5: Multi-feature cross-attention and Self-attention for the CFDF model.

$$
\begin{aligned}
Cross\_attention_1 &= softmax(\frac{Q_{CNN}K_{OP}^T}{\sqrt{d_k}})V_{OP} \\
Cross\_attention_2 &= softmax(\frac{Q_{CNN}K_{OF}^T}{\sqrt{d_k}})V_{OF}
\end{aligned}
\tag{5.6}
$$

where, Q_{CNN} (CNN feature modality), K_{OP} (OpenPose feature modality), V_{OP} (OpenPose feature modality) acts as the query, key, and value inputs, respectively, for the first cross-feature transformer and Q_{CNN} (CNN feature modality), K_{OF} (optical flow modality), V_{OF} (optical flow modality), respectively, for the second. Similarly, cross-attentions are calculated for the other multi-feature cross-attention block 2 and 3 with their respective base features.

The fused output from the cross-feature transformer block is passed over to the self-attention block (expanded and shown in Fig. 5.5) where the model learns a higher degree of attention between

the cross-feature attention blocks. Similar representations are obtained from the other cross-multi-feature attention blocks and self-attention blocks. The output from each of the self-attention blocks is further fused before passing through a final linear layer to learn the projections.

The output of the encoder which represents the relation between different features of the input video is passed as input to the multi-head attention block of the decoder thus learning the cross-attentions between the different sign features and the attentions from the words. The decoder takes in as input the word embeddings and performs masked-multi head attention by masking the future words. CFDF encoder output is fed to the multi-head attention block in the decoder where it learns the encoder-decoder attention and predicts the words after passing through a feed-forward network, linear, and softmax layers.

5.3.4 Training details

The TFLF and CFDF based transformer models are trained on 1027 ASL samples and 7096 GSL samples. Adam optimization is used with a learning rate of $1e^{-03}$ and a weight decay rate of $1e^{-03}$. The maximum length of frames in each batch is chosen as the input sequence length and the decoder is fixed at a maximum caption length of 30 based on the average length of the captions. The CFDF model has **128.17** million trainable parameters whereas the TFLF architecture has **57.57** million trainable parameters. The models have been trained for 70 - 150 epochs.

Architecture	Set	BLEU 1	BLEU 2	BLEU 3	BLEU 4
ResNet50 features only (R50)	Validation	24.75	16.04	11.56	8.95
	Test	23.67	14.58	10.29	8.00
OpenPose features only (OP)	Validation	24.82	15.87	11.53	9.07
	Test	23.52	15.24	11.00	8.67
Optical Flow features only (OF)	Validation	15.4	8.00	5.65	4.48
	Test	15.11	8.15	5.88	4.73
TFLF dim	Validation	**28.00**	**18.25**	**13.46**	**10.68**
	Test	**27.17**	**17.61**	**12.71**	**9.95**
CFDF dim	Validation	**28.95**	**19.05**	**13.91**	**11.03**
	Test	**27.33**	**18.18**	**13.26**	**10.46**

Table 5.1: Results on the GSL dataset using TFLF and CFDF transformer architectural setting

5.3.5 Performance Analysis

We test our dynamic cross feature dual model and tri-feature late fusion on the GSL dataset. We initially trained our model using only a single attention block belonging to individual features which mainly learns self-attention to understand how much each of the individual features contributes. Further, we train our CFDF multi-feature fusion model using all three feature inputs. Our CFDF and TFLF multi-feature fusion models perform better than the individual features by obtaining a $3-4$ points increase across BLEU 1-BLEU 4 scores as shown in Table 5.1. BLEU-BiLingual Evaluation Understudy [84] is a popular metric used to test the efficacy of predicted sentences with respect to the ground truth. n-grams (number of words) from predicted caption are compared with n-grams from the ground truth. Comparing individual words (1-gram) is referred to as BLEU 1, two words (2-gram) as BLEU 2, and so on.

Architecture	BLEU 1	BLEU 2	BLEU 3	BLEU 4
ResNet50 features only (R50)	10.70	4.67	3.21	2.66
OpenPose features only (OP)	10.23	6.79	5.8	5.36
Optical Flow features only (OF)	8.95	4.70	3.39	2.86
TFLF	14.62	8.97	7.18	6.31
TFLF (Transfer Learnt)	**22.16**	**16.81**	**14.67**	**13.55**
CFDF	13.98	7.64	5.60	4.59
CFDF (Transfer Learnt)	**22.39**	**15.96**	**13.56**	**12.25**

Table 5.2: Ablation study comparing the performance of the CFDF and TFLF multi-feature architecture with the single-feature architecture. The ablation study was performed on the low-resource ASLing dataset. The results show the BLEU 1 to BLEU 4 (B1, B2, etc) values; TL is the result of transfer learning from the GSL dataset, R50 - ResNet50, OP - OpenPose, OF - Optical Flow

We repeat the same experiments with the ASL dataset. From the results shown in Table 5.2, we can see that the TFLF and CFDF models perform similarly well as they are trained on less number of samples. To improve the performance we transfer learn from the best learned TFLF and CFDF models from the GSL dataset to the ASL dataset. Transfer learning shows significant improvement in the BLEU scores. The BLEU 4, and BLEU 1 scores improved by 7 and 8 points respectively, for the TFLF model and by 8 points for both BLEU 4 and BLEU 1 for the CFDF model. We see a 2 - 3 points boost in the BLEU 4 scores between the GSL and the transfer learnt ASL datasets for both the TFLF and CFDF models. We are expanding the current realistic ASL dataset so that transfer learning helps in boosting the BLEU scores considerably. Going forward, we will be using

the TFLF model since TFLF gives similar results as CFDF but is more memory efficient.

5.3.6 Attention visualization

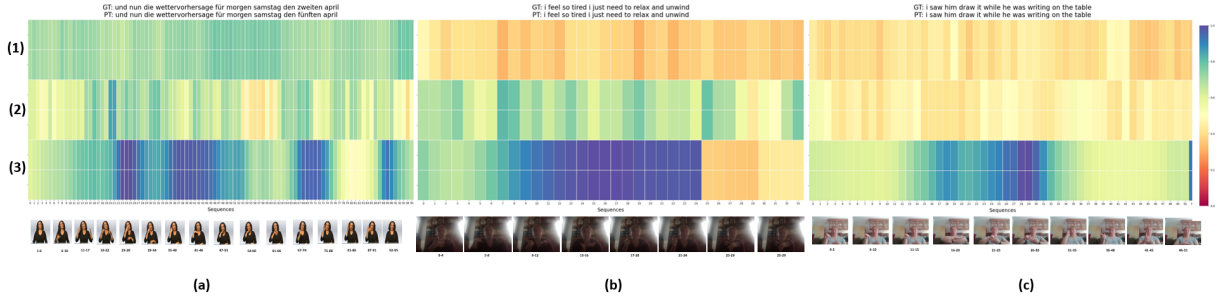


Figure 5.6: Attention visualization (best viewed in color). (a) best sample from the GSL dataset, (b), (c) best samples from the ASL dataset. (1) ResNet50 based fused features, (2) optical flow based fused features, (3) OpenPose based fused features. The samples chosen for visualization were one of the few where the BLEU 1 - BLEU 4 scores were close to 100%.

To understand the contribution of these three sets of features we looked at one of the test samples that gave the best BLEU scores for the GSL and the ASL dataset. The attention visualization is shown in Fig. 5.6.

The x-axis on the visualization are the different frames of the video under consideration and the y-axis is the combination of features. To visualize we store the attention weights from the last layer of each CFDF encoder and plot the heatmap against the sequence of frames.

Fig. 5.6 (a) shows the attention for one of the best test samples of the GSL dataset. From the frames, we can see that all the frames are pretty uniform in terms of the person annotating and the background. Thus, the model seems to be attending to ResNet50 based fused features almost equally in all the frames. The model attends to the optical flow based fused features where there is motion between the frames, for example, frames 13 - 32, 63 - 70, see a lot of changes between the consecutive frames leading to higher attention in these areas whereas, frames $47 - 61$ seem to have less motion between the signs leading to less attention. The model attends to OpenPose based fused features strongly for some sections of the video. Similarly, we picked a couple of samples from the ASL dataset which performed the best. From Fig. 5.6 (b) we can see that the frames under consideration are very dark in terms of the RGB visualization. The quality of the frames has a direct impact on the ResNet50 based fused features hence the model does not attend as

well when the image quality is poor. We see a similar pattern in both the datasets where the model attends fairly well to optical flow based features. We see a similar trend with OpenPose based fused features as well. We see similar trends in the other ASL sample as shown in Fig. 5.6 (c) where the image quality is low so the model does not attend as much to ResNet50 and optical flow based fused features.

Overall, we can see that the model often attends to OpenPose based fused features the most as OpenPose directly targets the hand, body, face keypoints which are the most essential for sign language. The model benefits from ResNet50 whenever there is a good quality video and benefits from optical flow whenever the flow movements between the frames are prominent. In a real-life scenario, there is no surety of having a good quality sign video, controlled image, prominent hand, and body movements, and/or face expressions. In situations like these, if only one of the features is used as input to the model, the output may not be accurate as the model could not attend to details based on the input feature. However, with fusion using CFDF, the model learns to attend to information from each of the features whenever and wherever applicable making the model robust to the changes in a real-life scenario.

5.3.7 Conclusion

In this work, we introduced the unconstrained ASLing dataset collected in real-world settings, where the participants could dress in their regular everyday clothes (with multiple textures and colors) and the signing videos were collected in arbitrary, unconstrained settings including the dorm rooms of DHH students. We focused on developing models to help us understand how best to perform high-quality ASL translation with multi-feature models, and in the absence of any gloss information.

To this end, we introduced the cross-feature dual-fusion (CFDF) architecture and provided it with multiple features (ResNet50 visual embeddings, OpenPose - body, hands, and face keypoints, and optical flow - frame-to-frame motion vectors) as inputs. We observed that this model dynamically attended to the ResNet50 features when the visual quality of the input frame was good, but the model attended more to the keypoints (body, hands, and face) for most of the frames. The model also attended to optical flow whenever there was a lot of movement and good flow-based information in the inputs. In our fusion work, we have focused on late fusion for both TFLF and CFDF

architectures rather than early fusion. For early fusion, we would typically concatenate all the features and then pass them through the attention models. But this defeats our purpose of finding which feature contributes the most.

Additionally, we discovered that we could successfully fine tune from a larger dataset (GSL) to boost the performance of the multi-feature fusion models on the ASLing dataset. This transfer learning paradigm significantly increased the resulting BLEU 1-4 scores on ASLing. To understand the inner workings of the models, we visualized the attention weights based on the three fused features and found that the model dynamically learned and attended to each of these features based on the input frame type.

In summary, in our research, we focus on improving the AI resources for ASL, which is still a low-resource language, in spite of all the resources readily available for its spoken/written counterpart.

5.4 Effects of feature scaling and fusion on SLT

To perform SLT, we introduce a multi-feature fusion architecture, shown in Figure 5.8. The model performs self-attention on individual features. The attention from individual features is then fed as input to the multi-feature attention (MFA) module which performs fused attention. The fused attention obtained from the encoder portion is then passed on to the decoder. The decoder takes as input the word embeddings, specifically the words predicted from previous time steps. The encoder output is used in the encoder-decoder attention block to attend to information between the visual features + pose and word embeddings. The decoder then outputs spoken language translation.

The subsections below describe various blocks of the multi-feature fusion architecture in detail.

5.4.1 Input Features

Pose features:

We extract pose features from OpenPose [48], an open-source toolkit for extracting body keypoints. We obtain x, y locations for 25 body-joint keypoints, 21 keypoints for each hand, and 70 facial landmark keypoints, resulting in 274 points per frame. We keep the order of the joints the same

as the original authors [48]. For frames where the lower body information is not available, the landmarks are zeroed out. We obtain a canonical form of the frame keypoints, by centering all points to the origin and scaling them to the same size. Examples of the canonical form of OpenPose points are shown in Figure 4.1 [1].

Scaled Features:

As sign language is challenging to segment, we aim to find implicit boundaries of signs by scaling the input frames. We apply a sliding window encompassing multiple frames across the video, resulting in overlapping video segments. The length of the sliding window can be 8, 12, or 16. The frames in each video segment are passed to the pre-trained Inception 3D CNN [108] model to obtain an output 1024 feature vector. This vector (or CNN embedding) is obtained for each video segment as shown in Figure 5.7. The features obtained from each of these three scaling mechanisms are fed to the multi-feature fusion model as shown in Figure 5.8.

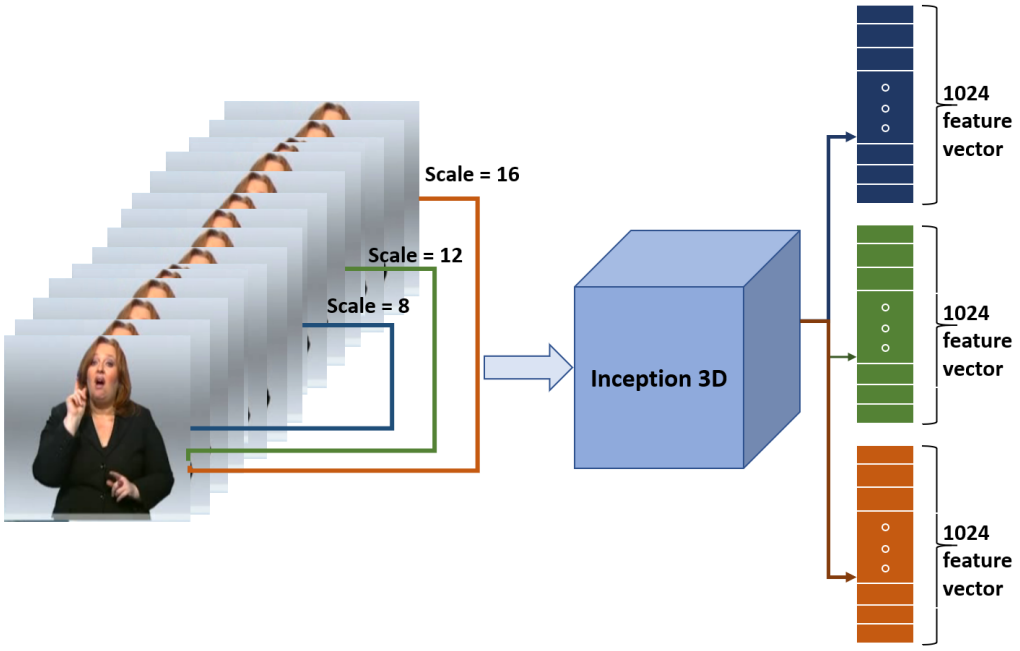


Figure 5.7: 3D CNN Feature scaling. Scale 8 (blue), Scale 12 (green), Scale 16 (orange) are shown (best viewed in color).

[1] These sample frames are not from the GSL dataset.

Consider X_1, X_2, X_3 to be the features processed by CNN_8, CNN_{12} and CNN_{16} as shown in Equation (5.7); where CNN_m is the i3D pre-trained CNN network whose inputs take m-frames.

$$\begin{aligned} X_1 &\in \text{CNN}_8\{x_1, x_2, .., x_n\} \\ X_2 &\in \text{CNN}_{12}\{x_1, x_2, .., x_n\} \\ X_3 &\in \text{CNN}_{16}\{x_1, x_2, .., x_n\} \\ X_4 &\in \text{OP}_8\{x_1, x_2, .., x_n\} \end{aligned} \tag{5.7}$$

Each x_i in $x_1, x_2, .., x_n$ represents a group of frames; i.e. x_1 for scale 8 is equivalent to frames $[0, 1, ...7]$, x_2 is equivalent to frames $[2, 3, ...9]$ and so on. For OpenPose we take the center frame which is a good representation for the whole segment.

Having overlapping frames in the segment is important, as the model learns better from the repetition of signs. Mirror padding is applied wherever needed so that the total number of frames for each video is the same for all three scales.

5.4.2 Multi-feature fusion architecture with scaled features

Encoder

The pose and visual features obtained from the scaling mechanism are shown in Equation (5.7). Here, X_1, X_2, and X_3 are features obtained through the implicit boundary segmentation processed by CNN_8, CNN_{12}, CNN_{16}, respectively.

Positional encoding is added to the input embeddings before passing it to the self-attention blocks. The positional encoding is essential to maintain the order information.

Each of the self-attention blocks ($S_{CNN8}, S_{CNN12}, S_{CNN16}, S_{OP8}$) compute the importance of each frame with respect to all the frames by initially performing a dot product between them (Q_{OP8}, K_{OP8}). The Softmax obtained from the dot product is then used to calculate the actual weight that each frame has in the original input by multiplying it again with the input frames (V_{OP8}). This

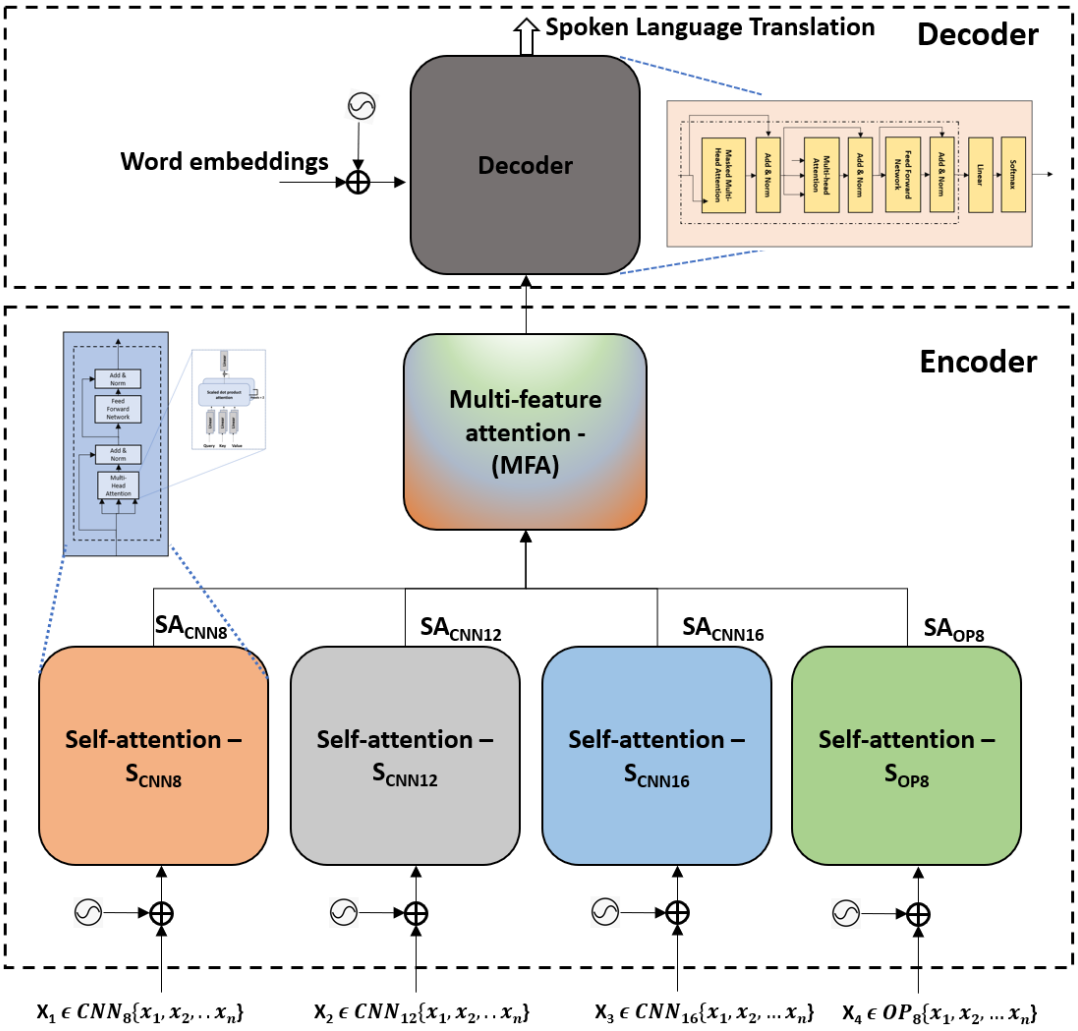


Figure 5.8: Multi-feature fusion architecture for sign language translation (best viewed in color). (Notations are described in Section 5.4.)

attention calculation for one of the blocks S_{OP8} is mathematically described in Equation (5.8).

$$\text{Self Attention } (SA_{OP8}) = \text{Softmax}(\frac{Q_{OP8}K_{OP8}^{T}}{\sqrt{d_k}})V_{OP8} \tag{5.8}$$

This attention is then passed on to a feed-forward network that learns two linear layers. Similar to the original implementation [23] we retain the add and norm blocks, and, skip connections.

The output from individual self-attention blocks is then fused and passed onto the MFA module that

now learns relations between the four features as shown in Equation (5.9).

$$\text{MFA} = (SA_{CNN8} \oplus SA_{CNN12} \oplus SA_{CNN_16} \oplus SA_{OP8}) \tag{5.9}$$

Decoder

The decoder begins to output predicted words upon receiving the start-of-sentence $< s >$ token. Only the words at the particular time-step are visible to the decoder. The embeddings are passed through a self-attention block for learning inter-relations between words. The decoder implements attention, feed-forward network, add and norm, and skip connections similar to the encoder. In addition to the above blocks, the decoder implements encoder-decoder joint attention in the multi-head attention block. The output obtained from the decoder is the spoken language equivalent of the sign language, thus performing continuous SLT.

5.4.3 Training details

The multi-feature fusion architecture is trained on 7096 samples from the GSL dataset. We obtained the best performance by implementing 3 layers in the encoder and decoder, along with 2 heads for multi-head attention in both the encoder and decoder.

During training, ground truth words were fed into the decoder at every time step and future words were masked out so that the model learning did not underfit. Whereas, during inference time, only predicted words at each time step are fed to the next time step, for a fair evaluation of the model. We selected an initial learning rate of $1e^{-04}$ and a weight decay rate of $1e^{-03}$. The model saturated after 80 epochs. We used an encoder embedding size of 256 and a custom decoder embedding size of $256 \times \#of features$.

We evaluated our models on the test set by calculating BLEU [84] scores. BLEU score is evaluated by comparing the predicted *n-grams* with the ground truth *n-grams*. BLEU 1 predicts *1-gram* words, i.e. it looks for matching individual words from predicted to ground truth. BLEU 1 scores are usually high due to this as the order of words does not matter. The task gets challenging as the model proceeds towards predicting longer grams. *2-gram* words compare two consecutive words,

3-gram compare 3 consecutive words, and so on. Because we are performing continuous SLT, BLEU 3 and 4 metrics are the most crucial for our evaluation.

Architecture	**BLEU 1**	**BLEU 2**	**BLEU 3**	**BLEU 4**
Conv2d-RNN [9]	27.10	15.61	10.82	8.35
Conv2d-RNN [9] + Luong Attention [22]	29.86	17.52	11.96	9.00
Conv2d-RNN [9] + Bahdanau Attention [78]	32.24	19.03	12.83	9.58
TSPNet-Sequential [109]	35.65	22.80	16.60	12.97
TSPNet-Joint [109]	**36.10**	**23.12**	16.88	13.41
Transformer (CNN_8) (Ours)	25.22	16.29	11.87	9.31
Transformer (CNN_{12}) (Ours)	24.48	16.01	11.71	9.31
Transformer (CNN_{16}) (Ours)	25.96	17.09	12.52	9.91
Transformer (OP_8) (Ours)	21.83	13.85	10.34	8.28
Multi-feature fusion (CNN_8, CNN_{12}, CNN_{16}) (Ours)	29.34	19.86	14.81	11.83
Multi-feature fusion (CNN_8, CNN_{12}, CNN_{16}, OP_8) (Ours)	33.59	23.07	**17.25**	**13.75**

Table 5.3: Results on the GSL dataset using the multi-feature fusion architecture with scaled input features.

Architecture	**BLEU 1**	**BLEU 2**	**BLEU 3**	**BLEU 4**
Conv2d-RNN [9]	27.10	15.61	10.82	8.35
Conv2d-RNN [9] + Luong Attention [22]	29.86	17.52	11.96	9.00
Conv2d-RNN [9] + Bahdanau Attention [78]	32.24	19.03	12.83	9.58
TSPNet-Joint [109]	**36.10**	**23.12**	16.88	13.41
Transformer (CNN_8) (Ours)	25.22	16.29	11.87	9.31
Transformer (CNN_{12}) (Ours)	24.48	16.01	11.71	9.31
Transformer (CNN_{16}) (Ours)	25.96	17.09	12.52	9.91
Transformer (OP_8) (Ours)	21.83	13.85	10.34	8.28
Multi-feature fusion (CNN_8, CNN_{12}, CNN_{16}) (Ours)	29.34	19.86	14.81	11.83
Multi-feature fusion (CNN_8, CNN_{12}, CNN_{16}, OP_8) (Ours)	33.59	23.07	**17.25**	**13.75**

Table 5.4: Results on the GSL dataset using the multi-feature fusion architecture with scaled input features.

5.4.4 Performance Analysis

To test the efficacy of our scaling mechanisms, we initially ran the single-input baseline transformer (on each of S_{CNN8}, S_{CNN12}, S_{CNN16}, S_{OP8}) using their respective individual features. From the results shown in Table 5.4 we can see that baseline scaled features perform better than some of the state-of-the-art methods like Conv2d-RNN, for BLEU 3 and 4. [9].

In addition to individual features, we perform a tri-feature experiment by passing only the three scaled CNN features through the multi-feature fusion architecture. Our model outperformed the best sequence model Conv2d-RNN, with attention [9] by achieving a 43.53% higher BLEU 4 score.

Our proposed model improves the translation results with the addition of OpenPose features and the 4-feature network outperforms the state-of-the-art [109] by achieving higher BLEU 3 and 4 scores.

5.4.5 Conclusion

In this work, we studied the effects of implicit boundary segmentation in signs and the effects of multi-feature fusion architecture for sign language translation without transcription. Using a sliding window approach we segmented the input video into multiple segments with overlapping frames. Our fusion architecture performed different levels of fusion, firstly it learned self-attention by learning relationships between different frames of the input video, and secondly, it performed multi-feature attention by learning the relations between the four sets of features.

Our model outperformed the recent state-of-the-art translation model by achieving higher BLEU 3 - BLEU 4 scores. It outperformed the state-of-the-art sequence networks with attention, by achieving a 43.53 % higher BLEU 4 score. Higher n-gram accuracy (BLEU 3, BLEU 4) indicates that the model is robust enough in predicting longer grams. In the future, we will consider benchmarking on larger datasets.

6. SignNet: Two-way Sign Language Interpretation using Metric Embedded Learning

6.1 Motivation

Two-way voice-controlled systems are becoming more popular with recent advances in technology. For example, Alexa by Amazon, Siri by Apple, Bixby by Samsung, etc. These systems have proven extremely beneficial for hearing and speaking, but not necessarily so for the Deaf-and-Hard-of-Hearing (DHH) community. In this work, we present a two-way sign language interpretation network.

The main intellectual merit of this work is the introduction of a novel learning mechanism, SignNet, consisting of several different aggregated losses, including a novel similarity metric-based loss, useful for 2-way sign language interpretation. We also present two paradigms for training, (i) a coupled model which jointly trains both the pose-to-text branch and the text-to-pose one, by propagating the losses through both branches in the entire network; and (ii) a decoupled model which trains each branch separately using its own predefined losses. We do this to determine if jointly training the coupled model has any performance improvements over the decoupled one.

We found the performance of the two training paradigms to be on-par, where one branch of the model performed better with decoupled training, while the other branch performed better in the coupled model. This implies that the two tasks are relatively independent and one branch of the model does not necessarily leverage the training information previously learned from the other branch.

The main societal benefit of this work is the presentation of a first attempt at a 2-way end-to-end sign language interpretation, useful, not only to the hearing-centric population (as is the case for

most SoTA AI models for continuous sign language translation) but also to the DHH community. This is important as it allows both the hearing and the DHH to converse freely with each, in their own first or preferred natural language. The model we present not only translates sign-poses to text but also takes as input text, and translates this back to sign-poses.

Importance of context learning in sign language translation

Sign language has some unique linguistic aspects that prove very challenging for automated sign translation, especially those that involve the use of the 3D space around the signer (referred to as the "signing space") [110]. While conversing, a signer will often associate places, people, and different entities with specific 3D locations close to her body. For example, a signer could finger-spell the name *J.a.n.e.t.*, the first time that person is mentioned in the conversation. The signer would now point to a location around her body and a spatial reference has now been created associating the person named *Janet* to that 3D location. In the course of the conversation, every instance in which the signer wants to refer to *Janet*, she will point to that location, the spatial reference point. In some instances, the signer may instead aim her gaze or tilt her head at the 3D location. As the conversation progresses and *Janet* is no longer in context, that reference point diminishes.

As another SL linguistic example, a signer may create spatial reference points, one on her left side for the subject and another on her right side for the object of the verb. In the course of the conversation, if there is a need to compare and contrast the subject and object, the signer may twist her waist and aim her torso to the right when discussing the object and then twist to the left for the object. This seeming exaggerated movement stresses which of the entities is being discussed and this is called *contrastive role shifting*.

There are significantly many more such 3D linguistic structures in sign language, where the context of the conversation is imperative in fully comprehending and translating the signs.

For these reasons, the ability to capture long-range contexts in signing, is critical to the success of 2-way sign language translation.

6.2 Related Works

Although the domain of video translation using deep learning is fairly new there have been few significant works that have changed the way videos are processed and translated. Sign Language Translation being one of them, is a complex problem that has required researchers a significant amount of time to come up with a decent solution. One such solution was proposed by Necati et al. [111] where a transformer model was built to recognize and translate sign language and was tested on RWTH PHOENIX-Weather-2014T (PHOENIX14T) dataset [112].

Many systems have been introduced and created to make it easier for deaf and hard of hearing people to communicate effectively using visual aids to produce signs based on fabricated sentences [113], [114], [115] . Different methods have been proposed to generate signs among which the avatar production [116], the Neural Machine Translation [117] have shown significant performance [118]. Due to their rather robotic appearance, the avatar systems did not receive much attention from the DHH community. To remedy this, Ebling et al. [119] gave it a more humanized form to eliminate the uncertainty of using sign language systems for effective communication.

In recent times, the quality of the production of signs has been enhanced using advanced neural network models. A system called Text2Sign was developed by Stoll et al. in which Generative Adversarial Networks were used to generate the sign videos. While Text2Sign had more appealing sign forms, Zelinka et al. [120] generated signs using a skeletal model based on the Openpose [121] framework. Each of these models presents sign generation per word in each sentence, Saunders et al. improve this by producing 3D continuous signs by utilizing gloss for human pose generation. This model was termed Progressive Transformer [122]. Using this Transformer model and Mixture Density Networks the same authors built a 3D multi-channel sign language production architecture [111].

6.3 SignNet: Two-way SLT

In our work, we present two-way SLT, SignNet, using two training paradigms, a coupled model and a decoupled model. Our SignNet architecture, as shown in Fig. 6.1, depicts the functioning of a coupled model. In the coupled SignNet we jointly train pose-to-text and text-to-pose networks.

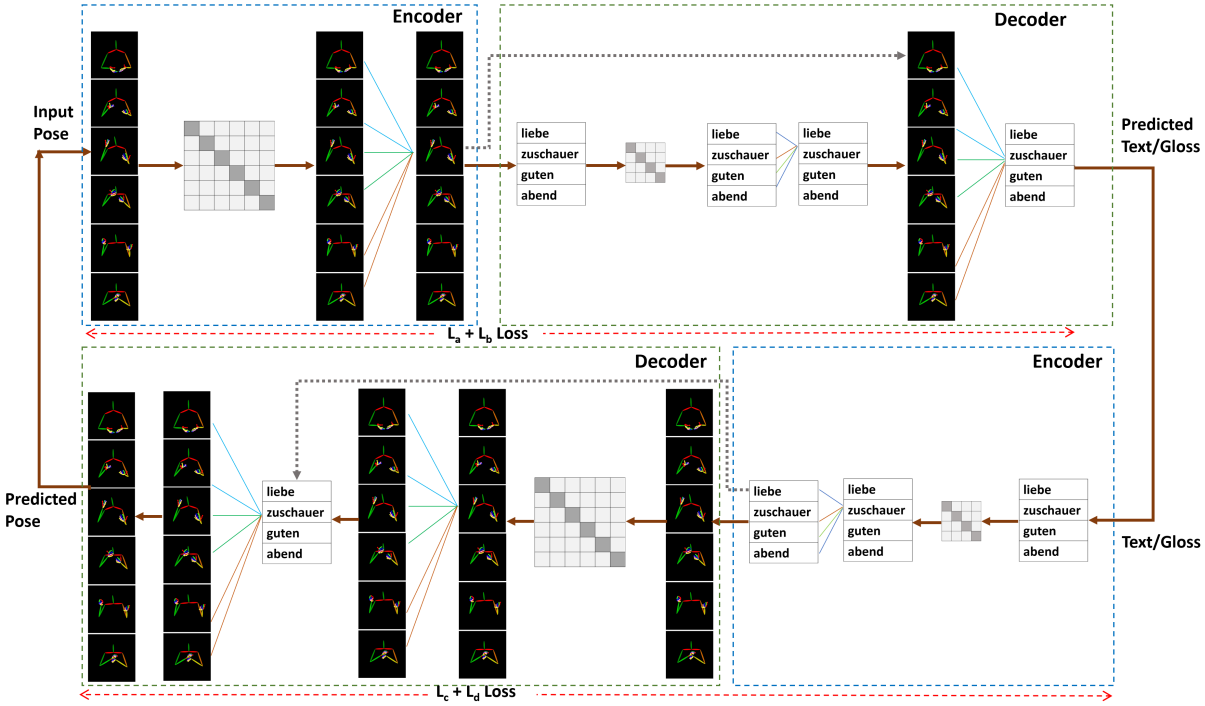


Figure 6.1: SignNet: Two-way Sign Language Translation. $L_a, L_b, L_c, and L_d$ are the losses used. Details in Section 6.3 (Best viewed in color).

Both the networks follow the encoder-decoder mechanism with different goals.

6.3.1 Coupled SignNet Pose to Text:

We obtain 3D pose features [120] by passing in 2D OpenPose joints [46] as input. Keeping pose generation in mind we mainly target hand joints, finger joints, and body joints until trunk. All these constitute 50 joint locations, thus giving us a vector of size 150 for each frame. The input can be denoted as $POSE = \{[(x_{0_1}, y_{0_1}, z_{0_1}), \ldots, (x_{149_1}, y_{149_1}, z_{149_1})], \ldots, [(x_{0_N}, y_{0_N}, z_{0_N}), \ldots, (x_{149_N}, y_{149_N}, z_{149_N})]\}$, where (x, y, z) represent the 3D joints and N is the number of frames. To retain the input ordering information we follow a similar pattern [23] and implement positional encoding for our input joints representation. For illustration purposes only, Fig. 6.1 shows how this positional encoding can be interpreted. Since sign language heavily depends on the context as explained in Section 6.1, we learn temporal dependencies by co-relating all the frames with respect to a single frame, continuously for all the frames, thus learning the context that is otherwise lost. To this end, we perform context learning between frames by initially computing the dot product of one

frame (Q) with all other frames (K) of the video under consideration. To avoid exploding values after dot product, we scale $\sqrt{d}$ [23] and then finally to retain the context information relevant to each frame, softmax activation ($softmax(\frac{QK^T}{\sqrt{d}})V$) is applied on these frames (V). These embeddings are then passed through a linear layer for improved features. We follow a similar pattern on the decoder side by initially obtaining word embeddings for each word, adding positional information, and then learning context between different words. Additionally, we learn the mapping between the frames and the words by taking the context information from the encoder and performing a scaled dot product with word-based attention. These embeddings are then passed onto a linear layer and softmax to predict continuous text or gloss.

6.3.2 Coupled SignNet Text to Pose:

The bottom part of Fig. 6.1 performs text to pose. The working of this block is similar to the pose-to-text explained above. Output from pose to text network is fed as input to the text to pose network. During training, we feed in actual information to facilitate better learning. The encoder portion now learns the context between different words of the sentence and the decoder portion learns the context between frames individually, and between frames and words. The output of this network is the respective pose which is fed back to the pose to text and the cycle continues until the learning saturates.

Since both the networks in coupled SignNet target different goals (good translation and good pose prediction) we implement four different loss functions explained in the next subsection and back-propagate the total loss throughout the whole network (coupled SignNet pose to text and coupled SignNet text to pose), thus successfully performing end-to-end learning for translation and pose prediction.

In the decoupled SignNet, we train the two networks showed in Fig. 6.1 individually. The working of these individual networks is the same as explained in the above sections.

6.3.3 Metric Embedded Learning for Pose Similarity

We are interested in ensuring that the predicted pose-based signs in the text-to-pose (T2P) arm of the 2-way SLT architecture predict continuous poses that are as similar as possible to the ground-truth

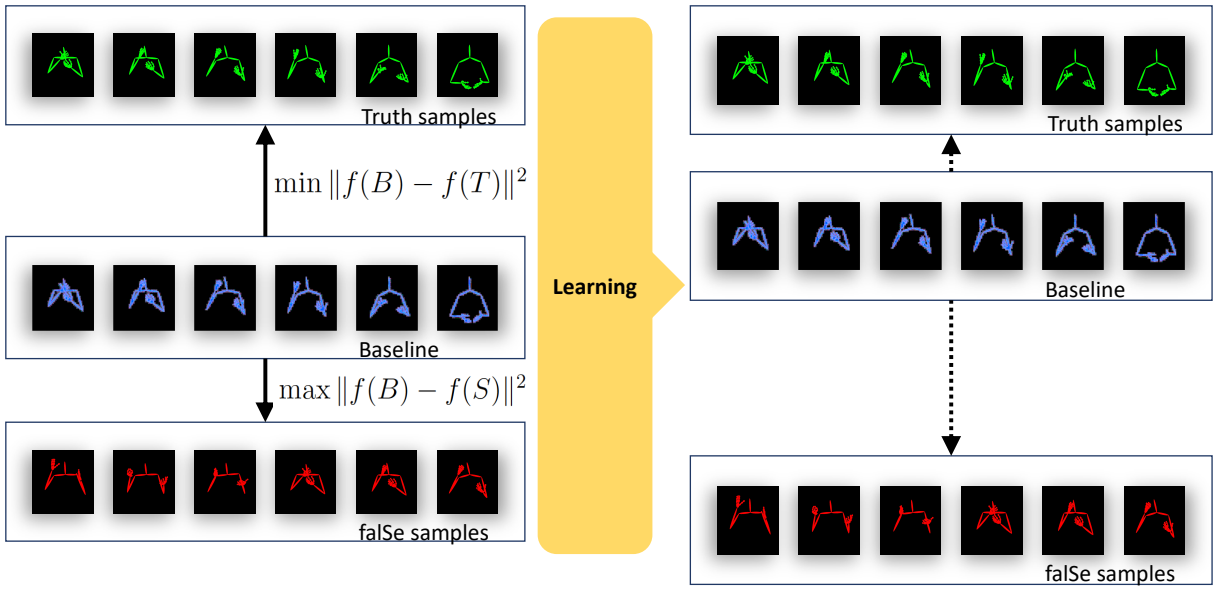


Figure 6.2: The loss based on sign similarity metrics minimizes the distance between a Baseline sign (ground-truth) and the Truth-like sign (the prediction for a sample-under-investigation), while maximizing the distance between the Baseline and a falSe sample(a different truth sign selected from the same batch as the sample-under-investigation). The LHS shows the the sets of samples before training and the RHS shows how similar signs are close, while different ones are far apart in the embedding space.

signs and as distant as possible to other signs in the same batch. See Fig. 6.2 provides a visualization of the desired mechanism.

To accomplish this, we can have

$$\underbrace{\|f(B) - f(T)\|^2}_{d(B,T)} - \underbrace{\|f(B) - f(S)\|^2}_{d(B,S)} \leq 0 \tag{6.1}$$

where B is a baseline sign, T is a truth sign required to be as similar to B as possible and S is a false sign (not as similar to the baseline); d(.) is the distance function.

To avoid the trivial solution where our function $f(.)$ will produce zero or one where $f(B) = f(T)$, similar to [123], we introduce a margin to impose a stronger constraint. The resulting distance function $d(B, T, S)$ is given in Equation 6.2:

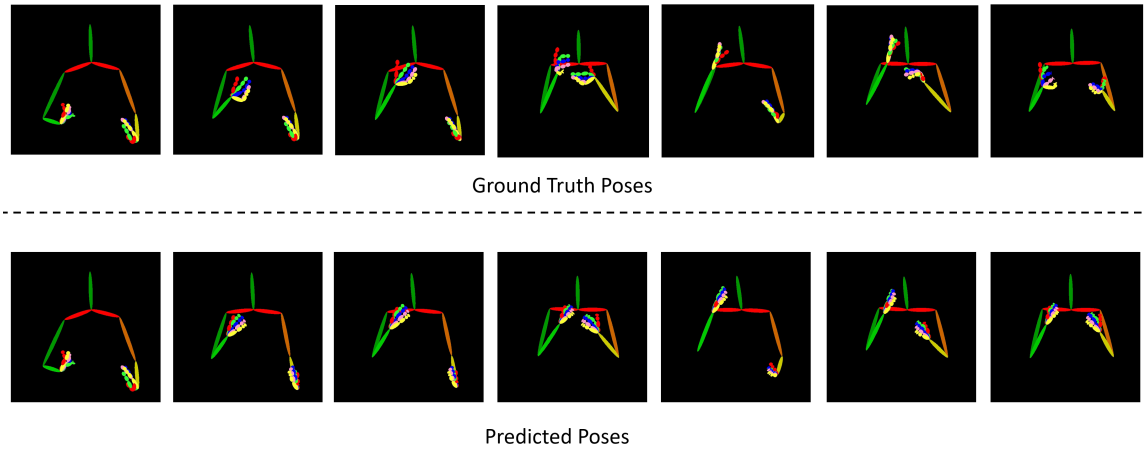


Figure 6.3: Ground Truth (Top) and Predicted (Bottom) poses using metric embedded learning. Test samples with lower loss and same time frames are chosen.

$$d(B, T, S) = \max(\ (d(B, T) - d(B, S) + \alpha), 0\ ) \tag{6.2}$$

We refer to the loss derived based on this distance as the *pose similarity metric-based loss function*, given in Equation 6.8, and is useful for enhancing the performance of the text-to-pose arm of the 2-way SLT training mechanism.

Choosing the similarity metric samples: While any random choice can readily satisfy $d(B, T) + \alpha \leq d(B, S)$, the underlying neural network will simply not learn if it gets it right too many times. But if the choice of samples is done such that $d(B, T) \approx d(B, S)$, the network is forced to work hard to learn the differences. This seemingly simple choice significantly increases the efficiency of the learning algorithm. We, therefore, select our samples in the following manner:

Consider a batch $\|B\| = 4$, where we are interested in calculating the similarity loss for the first sample $i = 1$. The baseline here is the ground-truth sign which we will refer to as $B^(i)$. The truth $T^(i)$ is the network prediction for sample i. Lastly, the false value $S^(i)$ is the ground-truth for any other sample $j \neq i \in \{B\}$, where j is randomly selected.

6.3.4 Loss functions for training SignNet

Recognition loss (L_a)

Gloss definition: *Gloss* is the written set of notations used to transcribe sign language into its written/spoken counterpart. Given that sign language is a visual-spatial language, in the absence of videos, gloss notations can be used to capture the sign-for-sign word ordering. It does this by providing different symbols useful for representing the spatial-temporal, facial, and 3D body grammar present in sign language. The grammar of a spoken/written language is very different from that of its sign language counterpart, where even the communication modalities are very different.

Gloss-driven recognition: For this reason, it is imperative to train the 2-way SL interpretation model using the recognition loss in addition to the translation loss described earlier. Recognition is an important intermediary step in the pose-to-text (P2T) branch of the 2-way SLT training mechanism.

Given the input sign video, as a sequence of sign poses $V = (s_1, \ldots, s_T)$ and the sequence of glosses $G = (g_1, \ldots, g_N)$ corresponding to V, the goal here is to learn $p(G|V)$. Because this sign to gloss mapping is monotonic and the word orderings are relatively consistent between signs and glosses, though requiring alignment between sequences of varying lengths, we employ the connectionist temporal classification loss, or CTC loss [124] for SL recognition.

CTC computes the loss between the unsegmented stream of input sign-pose embeddings and the target sequence of glosses. First, we obtain the pose-level gloss probabilities $p(g_t|V)$ by projecting the embeddings through a linear layer and softmax activation.

Then if we consider a path $\pi = (\pi_1, \cdots, \pi_T)$, the probability of a viable path given the video can be written as $p(\pi|V)$. CTC is thus used marginalize over the possible alignments of V to G, such that:

$$p_{ctc}(G|V) = \sum_{\pi \in \mathcal{M}} p(\pi|V) \tag{6.3}$$

where $\mathcal{M}$ is the set of viable paths in the sequence G;

Hence,

$$\mathcal{L}_a = \mathcal{L}_{ctc} = 1 - p(G^T|V) \tag{6.4}$$

where G^T is the ground-truth gloss sequence corresponding to video V.

Translation loss (L_b)

The primary task of the pose-to-text branch of the 2-way interpretation model is to generate a written/spoken language sentence $S = (w_1, \ldots, w_U)$ given a sign video V, as defined previously. The translation process discussed here aims to learn $p(S|V)$. Going from pose to sentences, in the decoding phase, we have:

$$p(S|V) = \prod_{i=1}^{U} p(w_i|w_{i-1}) = \prod_{i=1}^{U} \mathbf{Z}_{i,s_i} \tag{6.5}$$

where U is the length of the sentence and $\mathbf{Z} = (Z_{j,k}) = [z_1, \ldots, z_U]^\top$ is the probability distribution of the sentence when translated; $Z_{j,k}$ is the probability of word w_j having a word label k, given w_{j-1}.

$$\mathcal{L}_b = 1 - p(S^T|V) \tag{6.6}$$

where S^T is the ground truth sentence corresponding to video V, comprising of the aggregation of the ground truth probability of words during the decoding phase.

L_2 Regression loss (L_c)

The objective here is to learn the probability $p(V|S)$ of producing a sequence of sign-poses $V = (s_1, \ldots, s_T)$ over T time steps, given a spoken/written language sentence $S = (w_1, \ldots, w_U)$ having U words. Similar to the translation loss L_b described above, this L_2 regression loss is also computed from the output of the decoder, although this occurs in the text-to-pose branch of the model. Given the text sentence S as the inputs, the completed decoder output sequence of pose-signs can

be expressed as $\hat{s}_{1:T} = \hat{s}_1, \ldots, \hat{s}_T$. The Mean Squared Error (MSE) loss between the predicted sequence, $\hat{s}_{1:T}$, and the ground truth,$s^T_{1:T}$ is given as:

$$\mathcal{L}_c = \mathcal{L}_{LMSE} = \frac{1}{T}\sum_{i=1}^{t}(s^T_{1:T} - \hat{s}_{1:T})^2 \tag{6.7}$$

Sign similarity metric-based loss (L_d)

As shown in Figure 6.1, L_d is calculated at the text-to-pose arm of the 2-way mechanism and was introduced to reduce the network confusing similar signs. Details of the Sign similarity metric are given above in the section on gloss-driven recognition.

The sign similarity based loss over all samples can thus be given as:

$$\mathcal{L}_d = \sum_{i}^{M} d(B^{(i)}, T^{(i)}, S^{(i)}) \tag{6.8}$$

Justification: There is only a finite number of valid poses that make up a valid sign, hence there is often significant overlap between signs in the same batch. Without the strong constraint to separate truth and false examples, the network tends to readily confuse signs when predicting them from input text.

Total loss $\mathcal{L}$

The total loss for the 2-way end-to-end SLT architecture involves losses computed from the both the T2P and P2T branches of the model (as described above), and is given as

$$\mathcal{L}_{Total} = \lambda_a L_a + \lambda_b L_b + \lambda_c L_c + \lambda_d L_d. \tag{6.9}$$

In our experiments, using a grid search we found , we obtained the best performance with $\lambda_a = 5$, $\lambda_b = 5$, $\lambda_c = 100$ and $\lambda_d = 100$. Because the loss functions are computed differently, they required large ranges of weights to balance each other out, and give the best performance for the overall model.

6.4 Experiments and Results

6.4.1 Dataset

We evaluate our coupled and decoupled SignNet on one of the largest publicly available sign language datasets, the RWTH-PHOENIX-Weather dataset (RWTH),[1]. The sign videos in this RWTH dataset are based on German weather forecast airings. Each frame is of size 210×260 pixels with videos recorded at 25 frames per second (fps). We use the same split (7096/519/642, train/dev/test) as provided [1].

6.4.2 Metrics

To evaluate our SignNet pose to text translation we use BiLingual Evaluation Understudy (BLEU) [84] metric that evaluates how good the translation is by comparing the predicted text to its ground truth equivalent. BLEU 1 - BLEU 4 evaluates the performance based on 1-gram (individual words) to 4-gram words (group of consecutive four words). Additionally, we evaluate our text to pose SignNet network using Dynamic Time Warping (DTW) [125] metric that helps in aligning the poses as close to ground truth as possible. During coupled SignNet training we use DTW as our evaluation metric and optimize the end-to-end network for the lowest DTW score. For our decoupled SignNet networks, BLEU is used as the evaluation metric for the pose to text translation and DTW is used for text to pose translation.

6.4.3 Optimization and implementation

The input to SignNet coupled pose to text and decoupled pose to text is 3D poses generated by extracting 2D openpose [46] joints and then converting it to 3D [120]. We dropped facial joints and only choose body, hands, and fingers. Of all the body joints we only choose joints from head to the trunk. The ordering of chosen joints closely follows openpose [46] structure. Our coupled SignNet provides the best performance when using 7 encoder layers and 2 decoder layers(see Fig 6.1 top). We use an embedding dimension of 128. Similarly, for the bottom portion (in Fig. 6.1) of coupled SignNet, 2 encoder and decoder layers have shown optimal performance. We optimize our SignNet by using Adam optimizer [126]. We use a learning rate of 0.001 with plateau scheduling.

Experiment type	Set	BLEU 1	BLEU 2	BLEU 3	BLEU 4
PT G2P [122]	Dev	32.4	20.5	15.08	11.93
	Test	31.8	19.19	13.51	10.43
PT T2P [122]	Dev	31.41	19.97	14.80	11.82
	Test	31.36	19.04	13.54	10.51
2D Pose (using decoupled SignNet)	Dev	24.12	13.84	9.26	6.55
	Test	23.42	13.13	8.75	6.32
Coupled G2P (ours)	Dev	34.9	20.28	13.62	9.83
	Test	35.53	21.13	14.35	10.35
Coupled T2P (ours)	Dev	32.04	18.22	12.10	8.62
	Test	33.18	18.98	12.54	8.77
Decoupled G2P (ours)	Dev	**39.01**	**23.08**	**15.41**	10.98
	Test	**39.14**	**23.98**	**16.41**	**11.84**
Decoupled T2P (ours)	Dev	**37.05**	**22.14**	**15.04**	10.83
	Test	**36.76**	**21.78**	**14.77**	**10.66**

Table 6.1: Translation performance on predicted poses using coupled and decoupled SignNet. G2P - Gloss-to-Pose, T2P - Text-to-Pose.

Experiment type	Test			
	BLEU 1	BLEU 2	BLEU 3	BLEU 4
PT G2P [122]	31.8	19.19	13.51	10.43
PT T2P [122]	31.36	19.04	13.54	10.51
Decoupled G2P (ours w/o metric-based loss)	36.41	20.97	13.52	9.04
Decoupled T2P (ours w/o metric-based loss)	36.19	20.77	13.27	8.8
Decoupled G2P (ours w/ metric-based loss)	**39.14**	**23.98**	**16.41**	**11.84**
Decoupled T2P (ours w/ metric-based loss)	**36.76**	**21.78**	**14.77**	**10.66**

Table 6.2: Translation results using decoupled SignNet with (w/) and without (w/o) metric-based loss for sign similarity.

6.4.4 Discussion

We perform various experiments and compare our coupled and decoupled SignNet networks. Table 6.1 highlights the pose generation capabilities from our SignNet networks. We test coupled Gloss-to-Pose (G2P) and Text-to-Pose (T2P) by initially generating the poses on the dev set and test using the bottom part of the trained network described in Fig. 6.1. After the poses are generated, we pass the poses back as input through the top part of the trained network described in Fig. 6.1. This helps us in determining how good our predicted poses are. For decoupled SignNet, we obtain the pose predictions using the individual trained network and use the predicted poses to get the translations (translation model trained individually). In Table 6.1 we compare the performance

Experiment type	Test			
	BLEU 1	BLEU 2	BLEU 3	BLEU 4
Conv2d-RNN [9]	27.10	15.61	10.82	8.35
Conv2d-RNN [9] + Luong Attention [22]	29.86	17.52	11.96	9.00
Conv2d-RNN [9] + Bahdanau Attention [78]	32.24	19.03	12.83	9.58
Feature scaling (OP_8) [127]	21.83	13.85	10.34	8.28
TSPNet- Single 8 [128]	30.29	17.75	12.35	9.41
TSPNet- Single 12 [128]	29.02	17.03	12.08	9.39
TSPNet- Single 16 [128]	32.52	20.33	14.75	11.61
TSPNet-Joint [128]	36.10	23.12	16.88	13.41
Decoupled P2T (w/o metric embedded learning)	34.3	20.13	13.24	9.05
Coupled P2T for G2P (ours)	36.67	22.06	14.96	10.85
Coupled P2T for T2P (ours)	37.15	22.67	15.51	11.30

Table 6.3: Translation results using coupled and decoupled SignNet.

of our translations for predicted poses using our metric embedded learning mechanism with pose generated result [122] and decoupled 2D OpenPose. Our metric embedded learning boosts the performance of decoupled SignNet by achieving BLEU 1 score improvement 31.80/31.36 (G2P/ T2P) $\rightarrow$ 39.14/36.76 (G2P/ T2P) and 10.43/10.51 $\rightarrow$ 11.84/10.66 (G2P/ T2P) when compared with existing methods. We achieve significant improvement for BLEU 1 - BLEU 4 in comparison with [122]. To test how 2D pose points would perform we trained the decoupled SignNet with 2D pose generated from OpenPose [46] and compared it with our decoupled SignNet. We can see that decoupled G2P/T2P provides large improvements (BLEU 1: 23.42 $\rightarrow$ 39.14/36.76, BLEU 4: 6.32 $\rightarrow$ 11.84/10.66) when compared with 2D poses.

To verify that our metric embedded learning helps in improving the pose generation, we evaluate our decoupled model with and without metric embedded loss. Table 6.2 shows that our decoupled SignNet with metric embedding loss provides a good boost in performance by achieving higher BLEU 1 - BLEU 4 scores thus proving its efficiency.

Additionally, we tested how the text/gloss translation part of the network benefits from our coupled and decoupled SignNet networks using our combination of losses and metric embedded learning, respectively. In Table 6.3 we compare our translation results with existing single feature networks. While the first three results [9] use CNN features, the fourth one [127] uses pose features like ours. Our coupled SignNet boosts the BLEU 1 scores from 32.52 $\rightarrow$ 37.15, and significant improvements

Ground Truth	Predicted
G: liebe zuschauer guten abend	G: liebe guten abend
E: dear viewers good evening	E: love good evening
G: im süden freundliches wetter	G: im weht wetter
E: in the south friendly weather	E: in the blowing weather
G: und nun die wettervorhersage für morgen sonntag den fünften dezember	G: und die wettervorhersage für morgen donnerstag den achtzehnten juli
E: and now the weather forecast for tomorrow sunday the fifth of december	E: and the weather forecast for tomorrow thursday the eighteenth of july
G: der wind weht meist schwach aus unterschiedlichen richtungen	G: im weht schwach schwach aus unterschiedlichen richtungen
E:the wind usually blows weakly from different directions	E: im blowing weak weak from different directions
G: sonst ein wechsel aus sonne und wolken	G: im seltener aus sonne und wolken
E:otherwise an alternation of sun and clouds	E: in the seldom sun and clouds

Table 6.4: German translations using predicted pose generated from decoupled SignNet (G: German, E: English)

in BLEU 2 - BLEU 4.

When selecting our best SignNet for pose generation, Fig. 6.3 shows how our predicted pose closely aligns with the ground truth and in Table 6.4 we list out some of our best translations.

Between our coupled and decoupled SignNet, we expected joint training to improve performance, rather the change in the loss function made a significant impact by rendering better poses and translation. Our decoupled SignNet performs better than coupled SignNet because the latter is trained end-to-end to minimize the error towards pose generation. Due to this the sign to text translation network (upper half of Fig. 6.1) is not optimized enough and does not provide better results when evaluated on generated poses. However, decoupled SignNet individually optimizes each branch of SignNet thus providing better translations on the generated poses.

Limitations: The main limitation of our proposed SignNet model is its exposure to only a relatively small annotated training dataset, especially when compared with the larger datasets traditionally used in computer vision for similar video-to-text and text-to-video problems. Also, although we have developed a context-aware architecture, many aspects of sign language involving the long-range temporal dependencies among signs are still not well understood in current-day automated

SLT models. For example, it is not clear that a model such as proposed here can incorporate the unique but commonly-used SL linguistic spatial constructs such as the repeated use of spatial reference points, as described in Section 6.1.

6.5 Conclusion

In this chapter, we have presented SignNet, a 2-way sign language interpretation model which combines several different losses, in particular a novel similarity metric-based loss which significantly improves our translation performances. Because sign language has many linguistic aspects that involve the 3D space around a signer, which create the context for ensuing signs, SignNet is a context-aware network, that has successfully learned temporal dependencies by co-relating all the frames in an input sequence (whether input signs or input text) to each frame in the sequence.

To address 2-way translation, SignNet has two branches, one that converts text-to-pose and the other branch pose-to-text. In training the two branches, we explored the efficacy of jointly training (by back-propagating the losses from the more challenging branch, text-to-pose through the entire system) versus singly training the two branches. We found that the training paradigm did not make any real significant impact on performance, and the results obtained from both the coupled and decoupled models were on par. The main contribution to the performance boost was the inclusion of the similarity metric-based loss.

We demonstrate that SignNet predicts "good" signs when presented with input texts in scope, and also produces "good" text when presented with sign inputs. We show this by qualitatively examining predicted signs and compare them with their ground-truth counterparts. Similarly, we qualitatively compare predicted texts with the corresponding ground truth. When compared to similar state-of-the-art (SoTA) unimodal algorithms, SignNet yields by far the best results especially with respect to BLEU1-3 scores. BLEU4 scores are statistically on par with other SoTA algorithms when performing either of the 2-way tasks. Although there is still much to explore in our future work, we have successfully presented the first attempt at a 2-way end-to-end sign language interpretation, useful to both the hearing and the DHH population.

7. Conclusion

Automated sign language translation facilitates in creating a flexible and accessible communication platform between signing and non-signing people. This thesis contributed to the ongoing research in SLT by mainly focusing on improving the performance of SLT in a realistic environment by introducing real-world datasets and by proposing novel architectures that embark as the new SOTA models for SLT without any gloss. This thesis not only focused on SLT helping the hearing-centric but also an end-to-end model to facilitate both hearing and non-hearing populations by going from sign to text to sign translation.

In Chapter 2 we discussed various sign languages used in this work, German Sign Language (GSL), American Sign Language (ASL), Chinese Sign Language (CSL), and American Sign Language (ASLing - introduced). Although SL videos collected in a controlled environment with uniform background, illumination, and colors of clothes, like the GSL, helps in getting better quality results, this is not always possible. In real-world the sign language video could be very noisy with non-uniform background (clutter), a variety of colored clothes, poor illumination, etc (ASLing). Thus, in this thesis, we introduced ASLing collected in real-life settings. Additionally, we can conclude that since GSL is collected from weather airings, the vocabulary corpus isn't as widespread as needed to generalize the SOTA models. The embedding space between GSL and ASLing provides us with a baseline on how a realistic dataset should look like.

We studied how SLT has evolved from sequence modeling to transformer based modeling in Chapter 3. In the thesis, we performed various experiments using traditional models like recurrent neural networks, sequence-to-sequence modeling with and without attention, transformer modeling, sequence modeling with RL, and human evaluation. Based on the single feature SLT experiments in Chapter 4, we can conclude that the attention mechanisms in the transformer models have by far helped in boosting the SLT performance and thus created a good baseline for our future experiments. Additionally, we found that based on the type of video (quality, background, illumination, etc) the

performance of the sign languages varied. For videos with good quality, uniform background, and clothes, CNN features provided good information leading to better SLT. However, videos of poor quality with noise, pose-based features helped in extracting relevant information. Since in real-world it is not certain what type of video/frames will be encountered, we proposed a novel multi-feature fusion architecture in Chapter 5. We mainly targeted using our own ASLing dataset to evaluate the efficacy of our fusion architectures. Since ASLing is still a low-resource dataset we showed performance improvements on SLT by transfer learning from another high-resource SL dataset like GSL. Furthermore, as SL is challenging to segment we introduce feature-scaling by using 3D CNN features. We leverage our fusion architecture and create new SOTA results by beating the scores on popular datasets like GSL.

Finally, focusing only on improving sign-to-text is not enough as it only puts the emphasis on facilitating communication from the hearing-centric population. To facilitate two way communication between hearing and non-hearing centric people we introduced SignNet. SignNet jointly trains sign to text and text to sign using a metric embedded learning mechanism. To the best of our knowledge, we are one of the first ones to introduce such a complete system to perform end-to-end joint training and setting a benchmark by beating SOTA models and creating a new SOTA as explained in Chapter 6.

8. Future Work

In this chapter some prospective future work based on the current thesis are highlighted:

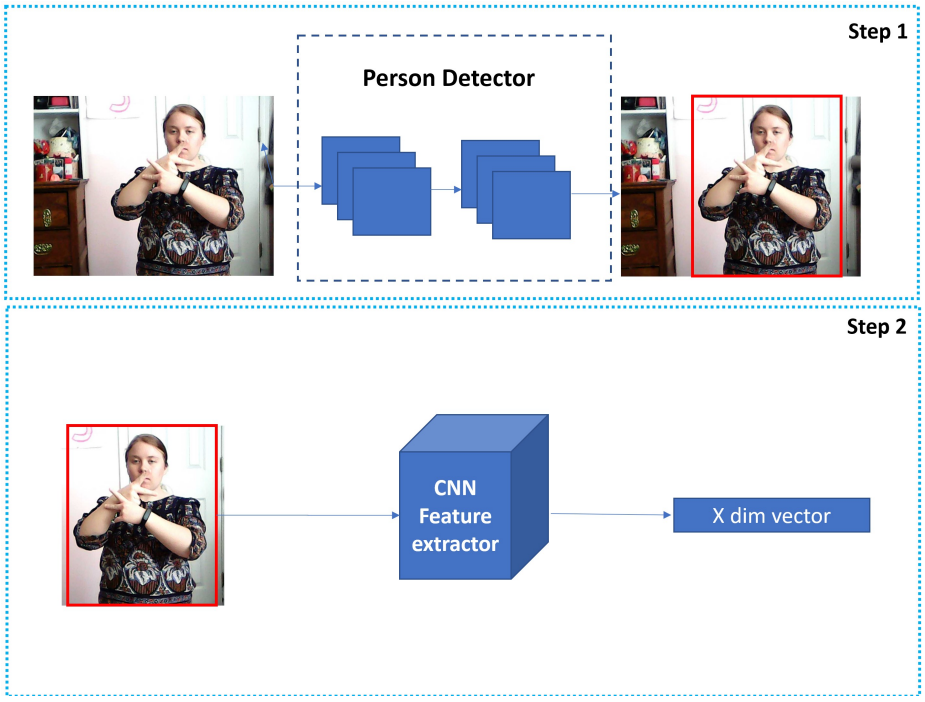


Figure 8.1: Person Detector

- **Implementation extensions:**

  Evaluate across different attention models, derived from - FairSeq [129], joeynmt [130], huggingface [131] to compare and contrast settings that bring out the best in SLT. The best setting can then be used in the multi-feature fusion and SignNet architectures.

- **Feature exploration:**

  1. Instead of using OpenPose to generate the body, hands, and face points, use a separate hands detector, face detector (E.g. OpenFace), body detector.
  2. Using generative adversarial networks (GANs) re-generate joints (body, hands, and face)

from input joints.

3. Two-step detection and SLT: This will be one of the first steps in prototyping for real-time SLT. In the two-step detection as shown in Fig. 8.1, initially detect the signer in the video and then extract CNN features required for SLT.

4. We are also interested in seeing how Unipose (2D and 3D) pose estimations [132, 133] aid SLT.

- **Expansion and application:**

  Our SignNet architecture can be further expanded towards novel applications like SonicASL [134]. When the asl signer signs, the doppler technology in the headphones senses the movements in the soundwaves and translates them to text.

- **Dataset Expansion:**

  ASL is still a low-resource dataset. We are currently analyzing 174 hours of ASL data scraped from the web. Potential next steps would be to try and run the multi-feature fusion architecture and SignNet on this dataset.

Figure 8.2: ASL data scraped from web

- **Conversational Sign Language:**

  Narrative (like storytelling) is very different from conversation. Much of the work in SLT falls under the narrative category. Little to no work on conversational SLT. SignNet is the first step towards conversational SLT. Instead of just narration, we would like to flow like a natural communication. Both parties communicate in their own natural mode of communication. There is a centerpiece that helps in quicker conversion from sign to text and text to sign and this is where our SignNet will be beneficial.

A. Select Publications

1. TACCESS: Deep Learning Methods for Sign Language Translation [135].
2. AAAI 2021 (Doctoral Symposium): A Computational Approach to Sign Language Understanding [136].
3. Interspeech 2021: Effects of Feature Scaling and Fusion on Sign Language Translation [137].
4. Face and gestures 2021 (Accepted): Dynamic Cross-Feature Fusion for American Sign Language Translation (accepted)
5. Face and gestures 2019 - Large scale sign language interpretation [3].
6. CVPR 2022:SignNet: Text to Sign Language Translation using Metric Embedded Learning (In submission).
7. TPAMI 2022: The role of transformers in SLT (Future submission).
8. IJCAI 2021: Exploring Effective Visual Features for American Sign Language Translation (Rejected).
9. ICMI 2021: Practical Sign Language Translation for a Variety of Input Signs (Rejected).
10. AAAI 2021: SignNet: Text to Sign Language Translation using Metric Embedded Learning (Rejected, revised, and submitted to CVPR 2022)

B. SignNet Ablations

In this chapter, some of the ablations performed to obtain the best settings for decoupled SignNet are shown in Table B.1. We performed experiments by varying the number of encoder and decoder layers and heads, varying the dimension, and by varying the recognition and translation weights. Red-colored row (Row 20) indicates the experiment that fetched us the best poses (G2P and T2P).

Exp No	Decoupled Experiment type	Test				Dev				Encoder			Decoder				
		B4	B3	B2	B1	B4	B3	B2	B1	# Layers	# Heads	Embd size	# Layers	# Heads	Embd size	Recognition Wt	Translation Wt
1	P2T	10.27	14.77	22.18	37.01	10.27	14.6	21.81	36.69	2	2	128	2	2	128	5	1
	G2P	7.51	11.17	17.49	30.71	8.15	11.75	18.04	31.08								
	T2P	7.71	11.45	17.74	30.83	7.69	11.24	17.59	30.53								
2	P2T	9.96	14.32	21.68	36.4	9.76	13.9	20.83	35.4	3	2	128	2	2	128	5	1
	G2P	7.25	10.88	17.35	30.82	7.17	10.61	16.7	29.94								
	T2P	7.04	10.63	17.14	30.68	6.63	9.93	15.93	29.32								
3	P2T	10.83	15.16	22.49	37.57	10.33	14.72	21.95	37.45	4	2	128	2	2	128	5	1
	G2P	4.1	7.02	12.99	25.86	4.54	7.48	13.21	25.47								
	T2P	3.79	6.66	12.6	25.11	4.05	6.91	12.9	25.43								
4	P2T	9.74	14.26	21.83	36.85	10.1	14.3	21.67	37.19	5	2	128	2	2	128	5	1
	G2P	8.15	12.27	19.16	33.5	7.72	11.57	18.49	33.68								
	T2P	7.86	11.9	18.72	33.17	7.97	11.66	18.43	33.32								
5	P2T	10.3	14.94	22.46	37.51	9.95	14.48	22.1	37.37	6	2	128	2	2	128	5	1
	G2P	7.59	11.34	17.55	30.49	7.32	11.12	17.6	31.36								
	T2P	7.53	11.16	17.08	30.07	7.55	11.26	17.58	31.15								
6	P2T	9.88	14.31	21.4	35.9	9.94	13.95	20.85	35.76	7	2	128	2	2	128	5	1
	G2P	8.37	12.58	19.56	34.21	8.58	12.65	19.89	35.09								
	T2P	8.28	12.5	19.71	34.75	9.07	13.27	20.5	35.67								
7	P2T	10.84	15.2	22.46	37.56	10.18	14.42	21.76	37.11	8	2	128	2	2	128	5	1
	G2P	8.13	12.05	18.51	31.97	8.19	11.94	18.55	31.99								
	T2P	8.21	12.02	18.55	31.81	7.64	11.31	17.88	31.51								
8	P2T	10.6	15.14	22.71	37.94	9.76	14.15	21.68	37.21	9	2	128	2	2	128	5	1
	G2P	8.22	12.16	18.82	32.95	7.92	11.73	18.35	32.76								
	T2P	8.11	11.98	18.68	32.88	7.71	11.45	18.02	32.34								
9	P2T	10.81	15.31	22.64	37.57	10.48	14.78	22.22	37.6	10	2	128	2	2	128	5	1
	G2P	7.44	11.06	17.17	30.33	7.04	10.6	16.93	30.74								
	T2P	7.38	11	17.03	30.05	6.93	10.34	16.61	30.59								

10	P2T	10.44	15.06	22.44	37.27	9.85	14.25	21.69	37.17	11	2	128	2	2	128	5	1
	G2P	8.3	12	18.42	32.64	7.85	11.59	18.08	32.64								
	T2P	8.17	11.96	18.5	32.69	7.58	11.47	18.19	32.73								
11	P2T	10.52	14.89	22.13	36.9	9.73	13.88	21.21	36.31	12	2	128	2	2	128	5	1
	G2P	7.46	11.21	17.59	30.9	6.95	10.47	16.82	30.58								
	T2P	7.18	10.79	17.1	30.61	7.25	10.83	17.2	30.85								
12	P2T	10.47	14.97	22.35	37.51	9.58	13.95	21.15	36.1	13	2	128	2	2	128	5	1
	G2P	7.78	11.54	18.02	31.53	7.88	11.69	18.19	31.93								
	T2P	7.58	11.46	18.26	31.87	7.88	11.78	18.4	32.17								
13	P2T	10.47	14.81	22.12	37.2	10.44	14.87	22.15	37.07	14	2	128	2	2	128	5	1
	G2P	6.27	9.46	14.97	27.13	7.08	10.45	16.23	27.99								
	T2P	6.25	9.38	14.87	26.94	6.79	9.98	15.62	27.36								
14	P2T	9.22	13.65	21.22	36.71	9.6	13.85	21.35	36.96	7	2	64	7	2	64	5	1
	G2P	8	12.09	19.39	35.21	7.18	11.2	18.48	34.33								
	T2P	7.92	12	19.3	35.11	7.13	11.05	18.25	34.36								
15	P2T	9.22	13.44	20.6	35.01	9.55	13.77	20.93	35.49	7	2	128	2	2	128	5	1
	G2P	8.79	12.95	20.04	34.58	9.07	13.39	20.58	35.19								
	T2P	8.39	12.51	19.67	34.08	8.85	13.06	20.27	34.91								
16	P2T	6.11	9.15	15.82	28.76	5.99	8.71	14	27.53	7	2	128	2	2	128	5	1
	G2P	6.1	9.14	14.79	28.47	5.99	8.69	13.94	27.14								
	T2P	6.1	9.14	14.8	28.48	5.94	8.63	13.88	27.11								
17	P2T	10.08	14.66	22.23	37.6	9.8	13.99	21.26	36.36	7	2	256	2	2	256	5	1
	G2P	7.35	11.33	18.14	32.03	7.58	11.49	17.14	31.83								
	T2P	7.38	11.5	18.45	32.39	7.81	11.58	18.11	31.81								
18	P2T	9.69	14.2	21.71	36.97	9.83	14.06	21.47	36.44	7	2	256	2	2	256	5	1
	G2P	7.05	10.59	16.67	29.47	6.71	10.15	16.32	29.61								
	T2P	7.13	10.69	16.72	29.46	6.91	10.26	16.17	29.21								
19	P2T	9.23	13.55	20.77	35.68	8.88	13.03	20.13	34.9	7	2	128	2	2	128	1	1
	G2P	8.6	12.69	19.52	34.04	8.21	12.14	18.95	33.13								
	T2P	8.86	12.96	19.72	34.08	8.09	12.09	18.71	32.66								
20	P2T	9.05	13.24	20.13	34.3	9.17	13.1	19.87	34.31	7	2	128	2	2	128	5	5
	G2P	9.19	13.69	21.24	36.6	9	13.39	21	36.79								
	T2P	9.06	13.52	21.07	36.62	9.08	13.39	20.89	36.51								
21	P2T	10.46	15.04	22.41	37.47	9.72	13.84	21.15	36.3	7	2	128	2	2	128	5	10
	G2P	7.36	10.98	17.45	32.3	7.21	10.89	17.84	32.46								
	T2P	7.14	10.74	17.28	31.85	7.33	11.12	17.85	32.71								
22	P2T	9.89	14.16	21.29	36.08	9.9	14.11	21.3	36.36	7	2	128	2	2	128	10	5
	G2P	7.95	11.9	18.54	32.88	7.44	11.44	18.66	33.73								
	T2P	7.63	11.7	18.44	32.76	7.91	11.65	18.52	33.35								
23	P2T	9.38	13.68	20.78	35.25	9.71	14.05	21.31	36.17	7	2	128	2	2	128	1	10
	G2P	8.75	12.9	19.94	35.08	8.28	1218	19.08	33.87								
	T2P	8.63	12.77	19.87	34.92	8.07	12.01	18.74	33.61								
24	P2T	8.8	13.06	20.23	35.15	9.03	12.95	19.74	34.22	7	2	128	2	2	128	10	10
	G2P	8.25	12.57	19.95	35.18	8.48	12.55	20	35.85								
	T2P	8.27	12.48	19.78	34.97	8.32	12.43	19.77	35.53								
25	P2T	9.93	14.43	21.84	36.62	10.22	14.45	21.8	36.82	7	4	128	3	2	128	5	1
	G2P	7.39	11.36	18.05	32.13	8.49	12.28	18.95	32.81								
	T2P	7.32	10.95	17.34	31.32	7.81	11.59	18.43	32.62								
26	P2T	10.78	15.22	22.53	37.15	9.95	14.13	21.46	37.04	7	8	128	3	2	128	5	1
	G2P	7.17	10.92	17.09	29.99	7.28	10.76	17.09	30.52								
	T2P	7.4	11.14	17.45	30.18	7.29	10.74	16.92	30.43								
27	P2T	10.08	14.45	22.01	37.34	9.86	14.04	21.39	36.88	7	8	128	4	2	128	5	1
	G2P	6.69	10.1	16.2	29.4	7.14	10.55	16.55	29.44								
	T2P	6.82	10.07	16.13	29.16	6.78	10.12	16.2	29.04								
28	P2T	10.29	14.86	22.41	37.2	10.4	14.8	22.1	37.46	7	8	128	5	2	128	5	1

	G2P	7.76	11.81	18.41	32.42	7.49	11.56	18.5	32.52								
	T2P	7.46	11.58	18.52	32.62	7.12	11.16	18.09	32.55								
29	P2T	10.11	14.63	22.28	37.67	9.86	14.23	21.75	37.18	7	8	128	6	2	128	5	1
	G2P	7.41	11.39	18.19	32.57	7.58	11.59	18.53	33.27								
	T2P	7.47	11.39	18.09	32.36	7.81	11.79	18.73	33.48								
30	P2T	10.55	15.13	22.4	37.38	9.56	13.84	21.1	36.51	7	8	128	7	2	128	5	1
	G2P	7.55	10.85	16.73	28.85	6.51	9.67	15.51	28.17								
	T2P	7.02	10.33	16.26	28.43	6.44	9.56	15.33	27.75								
31	P2T	8.84	13.18	20.53	35.5	8.53	12.73	20.09	35.17	7	8	128	8	2	128	5	1
	G2P	7.6	11.33	17.61	30.94	7.59	11.35	17.71	31.15								
	T2P	7.32	11.01	17.33	30.71	7.24	11.14	17.61	31.33								
32	P2T	10.11	14.5	21.77	36.63	10	14.53	21.85	36.56	7	2	128	2	4	128	5	1
	G2P	7.62	11.41	18.33	32.69	7.66	11.61	18.68	33.38								
	T2P	7.27	11.11	18.19	32.59	7.94	11.87	18.74	33.25								
33	P2T	10.23	14.58	21.87	36.86	9.54	13.74	21.07	36.44	7	2	128	2	16	128	5	1
	G2P	7.68	11.57	18.11	32.2	7.38	11.26	17.98	31.82								
	T2P	7.73	11.57	18.09	31.98	7.94	11.73	18.29	32.25								
34	P2T	10.64	15.17	22.8	37.87	10.35	14.66	21.93	36.89	7	2	128	2	8	128	5	1
	G2P	8.2	12.08	18.52	31.37	7.77	11.43	17.73	31.09								
	T2P	7.8	11.5	17.77	30.78	7.66	11.37	17.77	31.1								

Figure B.1: Decoupled SignNet ablations.

C. Single Feature (OpenPose) Ablations

We explore two different normalization techniques for OpenPose, min-max, and standardization. Using the architecture in Fig. 3.3 we evaluate OpenPose features on Min-Max and Standardization techniques. We also analyze how smoothing affects performance. From Table C.1 we can see that Standardization always provides the best results. Smoothing helps in improving the performance based on the organization of the dataset.

Table C.1: Ablation results using smoothing and normalization techniques on GSL, ASL, and CSL datasets. OP - OpenPose, AS - After Smoothing, BS - Before Smoothing, NN - Non Normalized features, MM - Min-Max Normalization, ST - Standardization technique. B1 - B4 repre-sent BLEU 1 - BLEU 4 scores.

		GSL				ASL				CSL			
Features	Set	B1	B2	B3	B4	B1	B2	B3	B4	B1	B2	B3	B4
OP (AS, NN)	Dev	23.13	14.64	10.58	8.25	-	-	-	-	-	-	-	-
	Test	21.35	13.01	9.11	7.04	11.78	6.82	5.15	4.3	26.81	21.99	20.45	19.67
OP (AS, MM)	Dev	17.27	9.80	6.78	5.23	-	-	-	-	-	-	-	-
	Test	16.24	8.95	6.08	4.64	9.02	4.76	3.37	2.77	14.92	7.80	6.25	5.57
OP (AS, ST)	Dev	**24.82**	**15.87**	**11.53**	**9.07**	-	-	-	-	-	-	-	-
	Test	**23.52**	**15.24**	**11.00**	**8.67**	10.23	6.79	5.80	5.36	43.03	38.93	37.52	36.76
OP (BS, NN)	Dev	22.18	13.40	9.63	7.54	-	-	-	-	-	-	-	-
	Test	20.62	12.17	8.39	6.35	11.40	6.30	4.57	3.77	27.47	22.38	21.01	20.17
OP (BS, MM)	Dev	21.94	13.50	9.55	7.37	-	-	-	-	-	-	-	-
	Test	19.19	11.34	8.02	6.17	7.40	3.63	2.66	2.23	15.01	7.77	6.20	5.55
OP (BS, ST)	Dev	24.20	15.28	11.08	8.69	-	-	-	-	-	-	-	-
	Test	22.69	14.61	10.38	8.09	**16.99**	**11.94**	**10.15**	**9.23**	**52.04**	**48.94**	**47.86**	**47.25**

In addition to OpenPose features, we extracted CNN features. These CNN features (ResNet-50, AlexNet, EfficientNet-B7, InceptionNet-V1, InceptionNet-V3) were pretrained on ImageNet [87]. We performed end-to-end (E2E) training with ResNet-50 and InceptionNet-V1. OpenPose and ResNet-50 features provide the best and similar results. OpenPose has a lighter memory footprint with 274 vector for each frame versus 2048 vector for each frame as in ResNet-50. For the evaluation of these results, we chose only the best performing features. The RGB quality of ASL and CSL datasets is poor and the datasets were collected in an unconstrained and uncontrolled environment. This is evident from the results in Table C.2. OpenPose features before smoothing and after standardization provide BLEU 1 - BLEU 4 of [16.99 - 9.23] for ASL versus [10.70 - 2.66] using ResNet-50. Similarly with CSL BLEU 1 - BLEU 4 scores fall in the range [52.04 - 47.25] with

OpenPose versus [13.20 - 4.02] with ResNet-50. This proves that OpenPose points help in boosting the performance even in scenarios when the input frame is noisy and not of good quality.

Table C.2: Experiments comparing OpenPose and CNN features on the GSL, ASL, and CSL datasets. OP - OpenPose, AS - After Smoothing, BS - Before Smoothing, ST - Standardization technique. B1 - B4 represent BLEU 1 - BLEU 4 scores.

		GSL				ASL				CSL			
Features	Set	B1	B2	B3	B4	B1	B2	B3	B4	B1	B2	B3	B4
OP (AS, ST)	Dev	**24.82**	15.87	11.53	**9.07**	-	-	-	-	-	-	-	-
	Test	23.52	15.24	11.00	8.67	10.23	6.79	5.80	5.36	43.03	38.93	37.52	37.76
OP (BS, ST)	Dev	24.20	15.28	11.08	8.69	-	-	-	-	-	-	-	-
	Test	22.69	14.61	10.38	8.09	**16.99**	**11.94**	**10.15**	**9.23**	**52.04**	**48.94**	**47.86**	**47.25**
ResNet-50	Dev	24.75	**16.04**	**11.56**	8.95	-	-	-	-	-	-	-	-
	Test	23.67	14.58	10.29	8.00	10.70	4.67	3.21	2.66	13.20	6.04	4.58	4.02

Bibliography

[1] Oscar Koller Hermann Ney Richard Bowden Necati Cihan Camgöz, Simon Hadfield. Rwth-phoenix-weather 2014 t: Parallel corpus of sign language video, gloss and translation. IEEE Conf. on Computer Vision and Pattern Recognition, Salt Lake City, UT, 2018, 05 2018.

[2] American Sign Language Dataset. `http://www.bu.edu/asllrp/`. [Online; accessed 9-May-2020].

[3] T. Yuan, S. Sah, T. Ananthanarayana, C. Zhang, A. Bhat, S. Gandhi, and R. Ptucha. Large scale sign language interpretation. In *2019 14th IEEE International Conference on Automatic Face Gesture Recognition (FG 2019)*, pages 1–5, 2019.

[4] World Health Organization. `https://www.who.int/news-room/facts-in-pictures/detail/deafness/`, 2018. [Online; accessed 04-Jan-2020].

[5] Markus eds. Hosemann, Jana; Steinbach. Atlas of sign language structures.

[6] Marcus Perlman, Hannah Little, Bill Thompson, and Robin L Thompson. Iconicity in signed and spoken vocabulary: a comparison between american sign language, british sign language, english, and spanish. *Frontiers in Psychology*, 9:1433, 2018.

[7] Five Paramters in Sign Language. `https://www.lifeprint.com/asl101/pages-layout/parameters.htm/`, [Online; accessed 24-March-2021].

[8] Roland Pfau and Josep Quer. Nonmanuals: Their prosodic and grammatical roles. *Sign languages*, pages 381–402, 2010.

[9] Necati Cihan Camgoz, Simon Hadfield, Oscar Koller, Hermann Ney, and Richard Bowden. Neural sign language translation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7784–7793, 2018.

[10] Wendy Sandler and Diane Lillo-Martin. *Sign language and linguistic universals*. Cambridge University Press, 2006.

[11] Glossing in Sign Language. `https://www.lifeprint.com/asl101/topics/gloss.htm`, [Online; accessed 23-April-2020].

[12] Glossing in Sign Language. `https://www.startasl.com/sign-language-symbols/`, [Online; accessed 23-April-2020].

[13] Necati Cihan Camgoz, Oscar Koller, Simon Hadfield, and Richard Bowden. Sign language transformers: Joint end-to-end sign language recognition and translation. *arXiv preprint arXiv:2003.13830*, 2020.

[14] Necati Cihan Camgoz, Oscar Koller, Simon Hadfield, and Richard Bowden. Multi-channel transformers for multi-articulatory sign language translation. *arXiv preprint arXiv:2009.00299*, 2020.

[15] O. Koller, N. C. Camgoz, H. Ney, and R. Bowden. Weakly supervised learning with multi-stream cnn-lstm-hmms to discover sequential parallelism in sign language videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(9):2306–2320, 2020.

[16] Nal Kalchbrenner and Phil Blunsom. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, 2013.

[17] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[18] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[19] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[20] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[21] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016.

[22] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025, 2015.

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[24] Alec Radford. Improving language understanding by generative pre-training. 2018.

[25] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.

[26] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. 2020.

[27] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[28] Subhashini Venugopalan, Marcus Rohrbach, Jeff Donahue, Raymond J. Mooney, Trevor Darrell, and Kate Saenko. Sequence to sequence - video to text. *CoRR*, abs/1505.00487, 2015.

[29] Haonan Yu, Jiang Wang, Zhiheng Huang, Yi Yang, and Wei Xu. Video paragraph captioning using hierarchical recurrent neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[30] L. Gao, Z. Guo, H. Zhang, X. Xu, and H. T. Shen. Video captioning with attention-based lstm and semantic consistency. *IEEE Transactions on Multimedia*, 19(9):2045–2055, 2017.

[31] Yingwei Pan, Ting Yao, Houqiang Li, and Tao Mei. Video captioning with transferred semantic attributes. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[32] Lorenzo Baraldi, Costantino Grana, and Rita Cucchiara. Hierarchical boundary-aware neural encoder for video captioning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[33] Silvio Olivastri, Gurkirt Singh, and Fabio Cuzzolin. An end-to-end baseline for video captioning. *CoRR*, abs/1904.02628, 2019.

[34] Nayyer Aafaq, Syed Zulqarnain Gilani, Wei Liu, and Ajmal Mian. Video description: A survey of methods, datasets and evaluation metrics. *CoRR*, abs/1806.00186, 2018.

[35] Xin Wang, Wenhu Chen, Jiawei Wu, Yuan-Fang Wang, and William Yang Wang. Video captioning via hierarchical reinforcement learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[36] L. Li and B. Gong. End-to-end video captioning with multitask reinforcement learning. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 339–348, 2019.

[37] Luowei Zhou, Yingbo Zhou, Jason J. Corso, Richard Socher, and Caiming Xiong. End-to-end dense video captioning with masked transformer. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[38] Ming Chen, Yingming Li, Zhongfei Zhang, and Siyu Huang. Tvt: Two-view transformer network for video captioning. In Jun Zhu and Ichiro Takeuchi, editors, *Proceedings of The 10th Asian Conference on Machine Learning*, volume 95 of *Proceedings of Machine Learning Research*, pages 847–862. PMLR, 14–16 Nov 2018.

[39] Chen Sun, Fabien Baradel, Kevin Murphy, and Cordelia Schmid. Contrastive bidirectional transformer for temporal representation learning. *CoRR*, abs/1906.05743, 2019.

[40] Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. Videobert: A joint model for video and language representation learning. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[41] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks, 2019.

[42] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *CoRR*, abs/1901.02860, 2019.

[43] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[44] Lionel Pigou, Aäron Van Den Oord, Sander Dieleman, Mieke Van Herreweghe, and Joni Dambre. Beyond temporal pooling: Recurrence and temporal convolutions for gesture recognition in video. *International Journal of Computer Vision*, 126(2-4):430–439, 2018.

[45] Noriki Nishida and Hideki Nakayama. Multimodal gesture recognition using multi-stream recurrent neural network. In *Image and Video Technology*, pages 682–694. Springer, 2015.

[46] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[47] Tomas Simon, Hanbyul Joo, Iain Matthews, and Yaser Sheikh. Hand keypoint detection in single images using multiview bootstrapping. In *CVPR*, 2017.

[48] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*, 2017.

[49] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *CVPR*, 2016.

[50] OpenFace. `https://github.com/TadasBaltrusaitis/OpenFace`, 2018. [Online; accessed 04-Jan-2020].

[51] Eldar Insafutdinov, Mykhaylo Andriluka, Leonid Pishchulin, Siyu Tang, Evgeny Levinkov,

Bjoern Andres, and Bernt Schiele. Articulated multi-person tracking in the wild. *CoRR*, abs/1612.01465, 2016.

[52] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[53] Petar Kormushev, Sylvain Calinon, and Darwin G Caldwell. Reinforcement learning in robotics: Applications and real-world challenges. *Robotics*, 2(3):122–148, 2013.

[54] Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.

[55] Jan Peters, Sethu Vijayakumar, and Stefan Schaal. Reinforcement learning for humanoid robotics. In *Proceedings of the third IEEE-RAS international conference on humanoid robots*, pages 1–20, 2003.

[56] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[57] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[58] AlphaGo Zero: Starting from scratch. `https://deepmind.com/blog/article/alphago-zero-starting-scratch`, [Online; accessed 1-May-2020].

[59] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*, 2015.

[60] Z. Zhang, J. Pu, L. Zhuang, W. Zhou, and H. Li. Continuous sign language recognition via reinforcement learning. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 285–289, 2019.

[61] Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. Self-critical sequence training for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7008–7024, 2017.

[62] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[63] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575, 2015.

[64] Haichao Shi, Peng Li, Bo Wang, and Zhenyu Wang. Image captioning based on deep reinforcement learning. In *Proceedings of the 10th International Conference on Internet Multimedia Computing and Service*, pages 1–5, 2018.

[65] Lijun Li and Boqing Gong. End-to-end video captioning with multitask reinforcement learning. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 339–348. IEEE, 2019.

[66] Boris Mocialov, Graham Turner, Katrin Lohan, and Helen Hastie. Towards continuous sign language recognition with deep learning. In *Proc. of the Workshop on the Creating Meaning With Robot Assistants: The Gap Left by Smart Devices*, 2017.

[67] Oscar Koller, Hermann Ney, and Richard Bowden. Deep learning of mouth shapes for sign language. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 85–91, 2015.

[68] Oscar Koller, Hermann Ney, and Richard Bowden. Deep hand: How to train a cnn on 1 million hand images when your data is continuous and weakly labelled. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3793–3802, 2016.

[69] Xiaoxu Li, Chensi Mao, Shiliang Huang, and Zhongfu Ye. Chinese sign language recognition based on shs descriptor and encoder-decoder lstm model. In *Chinese Conference on Biometric Recognition*, pages 719–728. Springer, 2017.

[70] Yuancheng Ye, Yingli Tian, Matt Huenerfauth, Jingya Liu, Nataniel Ruiz, Eunji Chong, James M Rehg, Sveinn Palsson, Eirikur Agustsson, Radu Timofte, et al. Recognizing american sign language gestures from within continuous videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 2064–2073, 2018.

[71] Lionel Pigou, Mieke Van Herreweghe, and Joni Dambre. Gesture and sign language recognition with temporal residual networks. In *Computer Vision Workshop (ICCVW), 2017 IEEE International Conference on*, pages 3086–3093. IEEE, 2017.

[72] Biyi Fang, Jillian Co, and Mi Zhang. Deepasl: Enabling ubiquitous and non-intrusive word and sentence-level sign language translation. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, pages 1–13, 2017.

[73] Shuo Wang, Dan Guo, Wen-gang Zhou, Zheng-Jun Zha, and Meng Wang. Connectionist temporal fusion for sign language translation. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 1483–1491, 2018.

[74] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376, 2006.

[75] Junfu Pu, Wengang Zhou, and Houqiang Li. Dilated convolutional network with iterative optimization for continuous sign language recognition. In *IJCAI*, pages 885–891, 2018.

[76] Necati Cihan Camgoz, Simon Hadfield, Oscar Koller, and Richard Bowden. Subunets: End-to-end hand shape and continuous sign language recognition. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[77] Sang-Ki Ko, Chang Jo Kim, Hyedong Jung, and Choongsang Cho. Neural sign language translation based on human keypoint estimation. *Applied Sciences*, 9(13):2683, 2019.

[78] KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259, 2014.

[79] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[80] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164, 2015.

[81] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[82] Kayo Yin. Sign language translation with transformers. *arXiv preprint arXiv:2004.00588*, 2020.

[83] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[84] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[85] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[86] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[87] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[88] Abraham Savitzky and Marcel JE Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical chemistry*, 36(8):1627–1639, 1964.

[89] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019.

[90] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.

[91] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[92] Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. Activation functions: Comparison of trends in practice and research for deep learning. *CoRR*, abs/1811.03378, 2018.

[93] R. E. Mitchell. How Many Deaf People Are There in the United States? Estimates From the Survey of Income and Program Participation. *Journal of Deaf Studies and Deaf Education*, 11(1):112–19, 2005.

[94] AmirAli Bagher Zadeh, Paul Pu Liang, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. Multimodal language analysis in the wild: CMU-MOSEI dataset and interpretable dynamic fusion graph. In *Proceedings of the 56th Annual Meeting of the ACL*, July 2018.

[95] Amir Zadeh, Paul Pu Liang, Navonil Mazumder, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. Memory fusion network for multi-view sequential learning. *arXiv preprint arXiv:1802.00927*, 2018.

[96] Yao-Hung Hubert Tsai, Shaojie Bai, Paul Pu Liang, J Zico Kolter, Louis-Philippe Morency, and Ruslan Salakhutdinov. Multimodal transformer for unaligned multimodal language sequences. In *ACL*, volume 2019, page 6558, 2019.

[97] Saurav Sahay, Eda Okur, Shachi H Kumar, and Lama Nachman. Low rank fusion based transformers for multimodal sequences. *arXiv preprint arXiv:2007.02038*, 2020.

[98] Yubo Jiang. Multi-feature fusion for video captioning. *IJCA*, 181(48):47–53, Apr 2019.

[99] Zekang Li, Zongjia Li, Jinchao Zhang, Yang Feng, Cheng Niu, and Jie Zhou. Bridging text and video: A universal multimodal transformer for video-audio scene-aware dialog. *arXiv preprint arXiv:2002.00163*, 2020.

[100] Jian Huang, Jianhua Tao, Bin Liu, Zheng Lian, and Mingyue Niu. Multimodal transformer fusion for continuous emotion recognition. In *ICASSP 2020-2020*, pages 3507–3511. IEEE, 2020.

[101] Umut Sulubacak, Ozan Caglayan, Stig-Arne Grönroos, Aku Rouhe, Desmond Elliott, Lucia Specia, and Jörg Tiedemann. Multimodal machine translation through visuals and speech. *Machine Translation*, 34(2):97–147, 2020.

[102] Katerina Papadimitriou and Gerasimos Potamianos. Multimodal sign language recognition

via temporal deformable convolutional sequence learning. *Proc. Interspeech 2020*, pages 2752–2756, 2020.

[103] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, volume 27, pages 568–576, 2014.

[104] Javier Sánchez Pérez, Enric Meinhardt-Llopis, and Gabriele Facciolo. TV-L1 Optical Flow Estimation. *Image Processing On Line*, 3:137–150, 2013.

[105] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Scandinavian conference on Image analysis*, pages 363–370. Springer, 2003.

[106] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. Vancouver, British Columbia, 1981.

[107] TVL1 OpenCV. `https://docs.opencv.org/3.3.0/dc/d47/classcv_1_1DualTVL1OpticalFlow.html`. [Online; accessed 11-Nov-2021].

[108] Joao Carreira, Pulkit Agrawal, Katerina Fragkiadaki, and Jitendra Malik. Human pose estimation with iterative error feedback, 2016.

[109] Dongxu Li, Chenchen Xu, Xin Yu, Kaihao Zhang, Ben Swift, Hanna Suominen, and Hongdong Li. Tspnet: Hierarchical feature learning via temporal semantic pyramid for sign language translation, 2020.

[110] Matt Huenerfauth and Pengfei Lu. Effect of spatial reference and verb inflection on the usability of sign language animations. *Universal Access in the Information Society*, 11(2):169–184, 2012.

[111] Ben Saunders, Necati Cihan, Camgöz, and Richard Bowden. Continuous 3d multi-channel sign language production via progressive transformers and mixture density networks. *CoRR*, abs/2103.06982, 2021.

[112] Oscar Koller, Jens Forster, and Hermann Ney. Continuous sign language recognition: Towards large vocabulary statistical recognition systems handling multiple signers. *Computer Vision and Image Understanding*, 141:108–125, 2015. Pose & Gesture.

[113] J. Glauert, R. Elliott, S. Cox, Judy Tryggvason, and M. Sheard. Vanessa - a system for communication between deaf and hearing people. *Technology and Disability*, 18:207–216, 2006.

[114] K. Karpouzis, G. Caridakis, S.-E. Fotinea, and E. Efthimiou. Educational resources and implementation of a greek sign language synthesis architecture. *Computers & Education*, 49(1):54–74, 2007. Web3D Technologies in Learning, Education and Training.

[115] John C. McDonald, Rosalee J. Wolfe, J. Schnepp, J. Hochgesang, D. Jamrozik, Marie

Stumbo, L. Berke, Melissa Bialek, and Farah Thomas. An automated technique for real-time production of lifelike animations of american sign language. *Universal Access in the Information Society*, 15:551–566, 2015.

[116] Michael Kipp, Alexis Heloir, and Quan Nguyen. Sign language avatars: Animation and comprehensibility. In Hannes Högni Vilhjálmsson, Stefan Kopp, Stacy Marsella, and Kristinn R. Thórisson, editors, *Intelligent Virtual Agents*, pages 113–126, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[117] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016.

[118] Razieh Rastgoo, Kourosh Kiani, Sergio Escalera, and Mohammad Sabokrou. Sign language production: A review. *CoRR*, abs/2103.15910, 2021.

[119] Sarah Ebling and Matt Huenerfauth. Bridging the gap between sign language machine translation and sign language animation using sequence classification. In *Proceedings of SLPAT 2015: 6th Workshop on Speech and Language Processing for Assistive Technologies*, pages 2–9, Dresden, Germany, September 2015. Association for Computational Linguistics.

[120] Jan Zelinka and Jakub Kanis. Neural sign language synthesis: Words are our glosses. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, March 2020.

[121] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *CoRR*, abs/1812.08008, 2018.

[122] Ben Saunders, Necati Cihan Camgöz, and Richard Bowden. Progressive transformers for end-to-end sign language production. *CoRR*, abs/2004.14874, 2020.

[123] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[124] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376, 2006.

[125] Donald J. Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, AAAIWS'94, page 359–370. AAAI Press, 1994.

[126] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[127] Tejaswini Ananthanarayana, Lipisha Chaudhary, and Ifeoma Nwogu. Effects of feature scaling and fusion on sign language translation. *Proc. Interspeech 2021*, pages 2292–2296, 2021.

[128] Dongxu Li, Chenchen Xu, Xin Yu, K. Zhang, Ben Swift, H. Suominen, and Hongdong Li. Tspnet: Hierarchical feature learning via temporal semantic pyramid for sign language translation. *ArXiv*, abs/2010.05468, 2020.

[129] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

[130] Julia Kreutzer, Jasmijn Bastings, and Stefan Riezler. Joey NMT: A minimalist NMT toolkit for novices. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, pages 109–114, Hong Kong, China, November 2019. Association for Computational Linguistics.

[131] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.

[132] Bruno Artacho and Andreas Savakis. Unipose+: A unified framework for 2d and 3d human pose estimation in images and videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021.

[133] Bruno Artacho and Andreas Savakis. Unipose: Unified human pose estimation in single images and videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[134] Yincheng Jin, Yang Gao, Yanjun Zhu, Wei Wang, Jiyang Li, Seokmin Choi, Zhangyu Li, Jagmohan Chauhan, Anind K. Dey, and Zhanpeng Jin. Sonicasl: An acoustic-based sign language gesture recognizer using earphones. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 5(2), June 2021.

[135] Tejaswini Ananthanarayana, Priyanshu Srivastava, Akash Chintha, Akhil Santha, Brian Landy, Joseph Panaro, Andre Webster, Nikunj Kotecha, Shagan Sah, Thomastine Sarchet, et al. Deep learning methods for sign language translation. *ACM Transactions on Accessible Computing (TACCESS)*, 14(4):1–30, 2021.

[136] Tejaswini Ananthanarayana. A computational approach to sign language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 15708–15709, 2021.

[137] Tejaswini Ananthanarayana, Lipisha Chaudhary, and Ifeoma Nwogu. Effects of feature scaling and fusion on sign language translation. *Proc. Interspeech 2021*, pages 2292–2296, 2021.